Titre: Title:	Problème de tournées de véhicules à horizon long : étude numérique d'une approche de stabilisation proximale
Auteur: Author:	Amar Oukil
Date:	2003
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Oukil, A. (2003). Problème de tournées de véhicules à horizon long : étude numérique d'une approche de stabilisation proximale [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7320/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/7320/
Directeurs de recherche: Advisors:	
Programme: Program:	Non spécifié

UNIVERSITÉ DE MONTRÉAL

PROBLÈME DE TOURNÉES DE VÉHICULES À HORIZON LONG : ÉTUDE NUMÉRIQUE D'UNE APPROCHE DE STABILISATION PROXIMALE

AMAR OUKIL
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
DÉCEMBRE 2003

Your file Votre référence
ISBN: 0-612-90850-X
Our file Notre référence
ISBN: 0-612-90850-X

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

PROBLÈME DE TOURNÉES DE VÉHICULES À HORIZON LONG : ÉTUDE NUMÉRIQUE D'UNE APPROCHE DE STABILISATION PROXIMALE

présenté par : OUKIL Amar
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a dûment été accepté par le jury constitué de :

M. DESAULNIERS Guy, Ph.D., président
M. SOUMIS François, Ph.D., membre et directeur de recherche
M. DESROSIERS Jacques, Ph.D., membre et codirecteur de recherche
M. ORBAN Dominique, Doctorat, membre

À mes très chers parents,

À Souad,

À la mémoire de Grand'mère,

Aux victimes des absurdités de l'homme.

REMERCIEMENTS

Si cette page paraît quelque peu banale parce que coutumière à toute thèse, je crois savoir aujourd'hui qu'elle correspond à un véritable besoin. Et si ***Merci****, comme* ***Bonjour****, me semblent bien souvent être des mots qui ne relèvent plus que de la bienséance, je voudrais que l'on retienne ici son acception la plus profonde.*

Je tiens à remercier MM. François Soumis et Jacques Desrosiers pour avoir bien voulu diriger ce travail et, surtout, pour l'opportunité qu'ils m'accordèrent de renouer avec la recherche.
À M. Hatem Ben Amor, je dis *MERCI.* Les orientations et les conseils qu'il n'a cessé de me prodiguer, sa constante disponibilité et sa patience sont autant de facteurs que, j'avoue, ont merveilleusement propulsé la concrétisation de cette étude. Une fois encore, *MERCI.*
Mes remerciements à MM. Guy Desaulniers et Dominique Orban pour l'intérêt qu'ils ont accordé à mon travail et pour avoir accepté de faire partie du jury de soutenance.

Autant, je dois remercier Mmes Suzanne Guindon, Francine Benoît, Nicole Paradis, Carole Dufour et bien d'autres collègues de Polytechnique et du GERAD qui sont restés constamment à l'écoute et dont l'aide fut très précieuse. Je ne saurais trop remercier MM. Abdelmoumène Toudeft et Pierre Girard qui m'ont immensément aidé sur bien des aspects informatiques.

Que mon épouse sache toute ma gratitude pour son aide et son soutien continus face aux problèmes divers qui entravaient la progression de mon travail.

Enfin, que ceux qui, comme Fabrice, Loïc, Abdoulaye, Ahmed et la liste est tellement longue, par leur présence et leur amitié, m'ont encouragé, en soient ici remerciés.

RÉSUMÉ

Le problème de tournées de véhicules multi-dépôts consiste à affecter une flotte de véhicules, localisés dans plusieurs garages, à un ensemble fini de parcours, de sorte à satisfaire des contraintes opérationnelles et techniques tout en utilisant de manière aussi efficace que possible les ressources disponibles. L'extension de l'horizon de temps des tournées donne lieu au problème de tournées de véhicules multi-dépôts à horizon long, un problème NP-difficile. La résolution du problème par génération de colonnes présente plusieurs aspects d'instabilité. La dégénérescence primale est responsable de l'effet de queue de la génération de colonnes. De plus, l'instabilité du comportement des valeurs des variables duales est plus fréquente et d'autant plus nocive lorsque l'horizon s'allonge. Au moment où la borne supérieure fournie par la solution primale est non-croissante, la borne inférieure calculée pour chaque point dual n'est pas nécessairement non-décroissante. Ainsi, un déplacement d'un bon point dual vers un autre plus mauvais est toujours possible, et ceci aura pour effet immédiat d'affecter la qualité des colonnes qui seraient générées à l'itération suivante. Par conséquent, l'objectif poursuivi dans ce mémoire consiste à développer une méthodologie d'approche basée sur la stabilisation du problème dual. Cette technique modifie le problème dual dans le but de guider l'évolution des multiplicateurs dans un sens favorisant la génération de colonnes de meilleure qualité. Nous étudions une approche de stabilisation proximale qui combine les concepts de région de confiance et de pénalisation linéaire par morceaux de l'objectif dual.

Nous avons d'abord entrepris la réduction de la taille du réseau. Il s'agit essentiellement de rechercher une borne inférieure au problème en résolvant sa relaxation en un problème de tournées de véhicules à dépôt unique ensuite, moyennant la solution primale ainsi obtenue, nous résolvons un problème de partitionnement pour trouver

une borne supérieure. Connaissant les coûts réduits associés aux colonnes de la solution primale, l'élimination des arcs *improductifs* est opérée conformément au principe de la fixation des variables. Outre la réduction du réseau, cette étape nous permet d'élaborer les outils nécessaires à la mise en œuvre de la stabilisation dont une solution initiale au problème dual. Préalablement à l'application de la stabilisation, nous avons élaboré un plan d'expérimentation pour tester la possibilité de généralisation des conditions initiales de la stabilisation à des problèmes de tailles et d'horizons différents. La stabilisation a été implantée en utilisant deux types de fonctions de pénalité linéaires par morceaux, plusieurs stratégies de mise à jour des paramètres de stabilisation et différents points pour l'initialisation des valeurs des variables duales. Conjointement, nous avons testé une stratégie d'accélération reposant sur la résolution du problème maître par la méthode duale du simplexe. Les calculs ont été effectués sur des problèmes générés aléatoirement sur des horizon allant d'une journée régulière à 7 jours. Les résultats obtenus illustrent parfaitement l'effet bénéfique de l'approche de stabilisation utilisée et montrent le potentiel d'une telle approche pour des applications pratiques.

ABSTRACT

The *Multiple-Depot Vehicle Scheduling Problem* (MDVSP) is to assign a fleet of vehicles, possibly stationed at several garages, to a given set of trips such that operational, technical and further side constraints are satisfied and the available resources are employed as efficiently as possible. The enlargement of the horizon of tours leads to the *long horizon* MDVSP, a NP-hard problem which is to be solved by a column generation method. Primal degeneracy, among other factors, is responsible for the tail effect of column generation. Moreover, instability in the behavior of the values of the dual variables is more frequent and harmful when the horizon gets longer. A lot of oscillations is observed and this gives raise to unuseful and even bad moves of high magnitude in the dual feasible space. Indeed, while the upper bound given by the primal solution is nonincreasing, the lower bound computed for each dual point is not nonincreasing. Hence, it is possible to move from a good dual point to a much worse one and this affects the quality of columns to be generated in the following iteration. Therefore, this thesis aims to developping a methodology to approach such an issue through a stabilisation of the dual problem. This technique modifies the dual problem to control and eventually guide the progress of the multipliers in a way that better quality columns are generated. We study a proximal type method that combines penalty and trust region concepts.

We first propose an approach to reduce the network size. The approach solves the SDVSP relaxation of the original MDVSP to compute a lower bound then, using the SDVSP primal solution, a set partitionning problem is solved to find an upper bound. The *unproductive* arcs are then dropped according to the variable fixing principle. Furthermore, an initial dual solution, necessary to the stabilization process, is produced. Before undertaking the stabilization, we worked out an experimenting

plan to allow a generalization to different size and horizon length problems of the initial conditions required to launch the stabilization procedure. The stabilization is implemented using two types of piecewise linear penalty functions, many strategies to update the stabilization parameters and different vector values to initialize the dual variables. Jointly, we tested an accelerating strategy based upon solving the master problem with the dual simplex method. Computational experiments were carried out over randomly generated multi-depot vehicle scheduling instances of 1 day to 7 day length horizons. The outcome shows a great stabilizing and accelerating effect of column generation method.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ANNEXES

INTRODUCTION

Du commencement, on peut augurer la fin
Quintilieu (De l'instititione oratoria)

La résolution des problèmes de transport a été et reste toujours l'une des forces motrices du développement des disciplines mathématiques comme l'optimisation et la recherche opérationnelle [4]. Des problèmes de transport de grande taille doivent être résolus, par exemple, en traffic aérien (construction des itinéraires d'avions et des rotations d'équipages) et en transport public (confection de tournées de véhicules et affectation des tâches). Au cours des quelques dernières décennies, la planification des grands systèmes de transport public a été subdivisée en un processus hiérarchique, impliquant la planification des routes (*line planning*), la confection des horaires (*timetable planning*), la confection des tournées de véhicules (*vehicle scheduling*) et celle des rotations d'équipes de travail (*duty scheduling and rostering*). La résolution de chacune de ces étapes est en soit une tâche délicate. Notre intérêt portera sur le problème de confection de tournées de véhicules et plus explicitement, sur le cas multi-dépôts à horizon long.

Le problème de tournées de véhicules multi-dépôts consiste à affecter une flotte de véhicules, localisés dans plusieurs garages, à un ensemble fini de parcours, de sorte à satisfaire des contraintes opérationnelles et techniques tout en utilisant de manière aussi efficace que possible les ressources disponibles. Ce problème, réputé NP-difficile, continue de susciter des efforts considérables pour développer des techniques à même de le résoudre en des temps CPU raisonnables (voir, par exemple, [5], [7] et [9]). L'extension de l'horizon des tournées à plusieurs jours au lieu d'une journée nous ramène au problème de tournées de véhicules multi-dépôts à horizon long, un problème encore difficile dont nous nous assignons la tâche de résoudre. La résolution

du problème par génération de colonnes présente plusieurs aspects d'instabilité. La dégénérescence primale est responsable de l'effet de queue de la génération de colonnes. De plus, l'instabilité du comportement des valeurs des variables duales est plus fréquente et d'autant plus nocive lorsque l'horizon s'allonge. Au moment où la borne supérieure fournie par la solution primale est non-croissante, la borne inférieure calculée pour chaque point dual n'est pas nécessairement non-décroissante. Aussi, un déplacement d'un bon point dual vers un autre plus mauvais est toujours possible, et ceci aura pour effet immédiat d'affecter la qualité des colonnes qui seraient générées à l'itération suivante. Ainsi, l'objectif poursuivi dans ce mémoire consiste à développer une méthodologie d'approche basée sur la stabilisation du problème dual. Cette technique modifie le problème dual dans le but de guider l'évolution des multiplicateurs dans un sens favorisant la génération de colonnes de meilleure qualité. Nous étudions une approche de stabilisation proximale qui combine les concepts de région de confiance (*trust region*) et de pénalisation linéaire par morceaux de l'objectif dual. A cet effet, nous avons structuré le présent mémoire en cinq chapitres.

Un premier chapitre est consacré au problème de tournées de véhicules multi-dépôts, lequel est défini de manière explicite en présentant les deux formulations communément rencontrées dans la littérature. Ensuite, nous passons au problème qui fait l'objet de notre étude : le problème de tournées de véhicules multi-dépôts à horizon long. Pour mettre l'accent sur les aspects spécifiques du problème, nous commençons par décrire l'algorithme qui permet de générer des instances de problèmes test relatives à des horizons de temps s'étendant d'une journée régulière à une semaine (7 jours). En nous basant sur la composition des réseaux ainsi générés, nous anticipons certaines conclusions en rapport avec la résolution des problèmes de tournées de véhicules à horizon long.

Au deuxième chapitre, nous montrons à quel point l'extension de l'horizon peut avoir des répercussions néfastes sur les temps de résolution des problèmes de tournées de

véhicules multi-dépôts par la méthode de génération de colonnes. Des batteries de problèmes test sont résolus. Au travers des résultats obtenus, une première étape sera franchie en vue de cerner la problématique et, de là, envisager des outils mieux appropriés pour y pallier. Il est clair que la difficulté augmente à mesure que l'horizon s'allonge, au point où des problèmes, tels ceux d'un horizon de 7 jours, restent insolubles bien qu'on leur ait alloué un temps de calcul dépassant *5 mois*.

Le troisième chapitre se veut une étape charnière sur la voie de la résolution du problème. Nous exposons l'algorithme de base d'une approche de résolution du problème de tournées de véhicules multi-dépôts, abstraction faite de l'aspect horizon long. L'idée maîtresse de cet algorithme va dans le sens de la réduction de la taille du réseau associé au problème, cela préalablement à sa résolution. Il est question de rechercher une borne inférieure au problème en résolvant sa relaxation à un seul dépôt avant de procéder à la résolution d'un problème de partitionnement, moyennant la solution primale ainsi obtenue, pour trouver une borne supérieure. Connaissant les coûts réduits associés aux colonnes de la solution primale, l'élimination des arcs *improductifs* est opérée conformément au principe de la fixation des variables (*variable fixing*). Nous expliquons les extensions possibles de cet algorithme en énumérant les plus-values qu'elles peuvent générer. Grâce à cette approche, l'opportunité nous est offerte d'enrichir la plateforme de la problématique en y incluant des éléments techniquement plus objectifs.

Un quatrième chapitre porte sur l'environnement théorique nécessaire à la mise en œuvre de la stabilisation de l'algorithme de génération de colonnes par une fonction de pénalité linéaire par morceaux. Une brève revue de littérature est présentée avant de détailler les aspects théoriques pertinents à l'étude en cours. Nous nous intéressons exclusivement aux techniques de stabilisation de l'algorithme de génération de colonnes basées sur les concepts de région de confiance (*trust region*) et de pénalisation linéaire par morceaux de l'objectif dual. Nous proposons des extensions de certains

points, notamment la structure de la fonction de pénalité et les stratégies de mise à jour des paramètres de stabilisation.

Pour terminer, le dernier chapitre retient l'essentiel des développements effectués le long de ce travail. Nous présentons les résultats numériques les plus importants dont le plus frappant est la résolution des problèmes sur un horizon de 7 jours en moins de *2* minutes, ceux-là mêmes qui étaient difficiles à résoudre en *5* mois avec la procédure de génération de colonnes standard. Le long de cette étude, nous ouvrons, à chaque fois que l'opportunité se présente, des brèches pour des travaux futurs.

CHAPITRE 1 : PROBLÈME DE TOURNÉES DE VÉHICULES À HORIZON LONG

Étant donné une flotte de véhicules disponibles à différents dépôts, le problème de tournées de véhicules multi-dépôts (*MDVSP : Multiple Depot Vehicle Scheduling Problem*) ([13], [23], [28]) consiste en l'élaboration des itinéraires permettant d'accomplir, au moindre coût, un ensemble de tâches définies dans l'espace et le temps. Des problèmes de ce type sont fréquents, entre autres, en transport urbain. Pour le cas particulier du transport par autobus, les tâches sont des parcours d'autobus et l'objectif est la construction des tournées optimales pour des véhicules en provenance de garages situés en des points divers d'une localité géographique. Les tournées sont composées d'une suite de parcours entrecoupés de déplacements à vide [20].

Le *MDVSP* est une version de base de la classe des problèmes de confection de tournées et d'horaires pour lesquels d'autres aspects peuvent être inclus, notamment les fenêtres de temps et les contraintes de ressources. Dans le cas de notre étude, nous axerons nos efforts sur ce problème sous sa forme la plus élémentaire avant d'en envisager d'éventuelles extensions. À cet effet, nous commençons par en présenter les formulations communément utilisées dans la littérature. Celles-ci se présentent essentiellement comme des programmes en nombres entiers dont la structure dépend de la définition des variables de décision. Ainsi, deux formulations sont fréquemment rencontrées, l'une adopte une approche multiflots utilisant des variables de flot sur les arcs, tandis que l'autre fait intervenir des variables de flot sur les chemins. Les deux approches seront détaillées dans les sections ci-après. Par la suite, nous décrirons la particularité du problème de tournées de véhicules multi-dépôts à horizon long.

1.1 Problème *MDVSP*

Soit un ensemble de tâches $\{T_1, T_2, \ldots, T_n\}$ données a priori. Chaque tâche T_i est un trajet décrit par un point de départ et un point d'arrivée et commence au temps a_i et se termine au temps b_i. Il s'agit de couvrir toutes ces tâches en utilisant les véhicules disponibles aux dépôts D_k ($k \in K$), abritant chacun n_k véhicules. Les véhicules sont considérés identiques.

Le problème est de trouver une affectation optimale des tâches aux véhicules, telle que :

- chaque tâche T_i est couverte par exactement un véhicule,
- le nombre total de véhicules partant de chaque dépôt D_k ($k = 1, 2, \ldots, |K|$) et utilisés dans la solution ne dépasse pas la capacité n_k de ce dépôt,
- chaque véhicule qui part du dépôt D_k retourne au même dépôt après avoir effectué sa tournée,
- le coût total associé aux tournées de véhicules produites par la solution doit être minimal (les véhicules non utilisés ne contribuent pas au coût).

La solution du problème détermine le nombre de véhicules à utiliser ainsi que leurs itinéraires respectifs.

1.1.1 Description du réseau associé au problème

Désignons par G^k=(V^k, A^k) le réseau associé au dépôt D_k ($k = 1, 2, \ldots, |K|$) où V^k représente l'ensemble des nœuds et A^k l'ensemble des arcs. Les nœuds du réseau sont les tâches et les dépôts. En l'occurrence, chaque ensemble V^k contient deux types de nœuds : n nœuds-tâches qui forment l'ensemble N et un nœud dépôt que l'on identifiera par $n + k$, de sorte que $V^k = N \cup \{n + k\}$.

Si t^k_{ij} est le temps de déplacement entre deux nœuds i et j du réseau G^k, le calcul de

t_{ij}^k diffère selon que ces nœuds sont des tâches ou des dépôts. Aussi, si i et j réfèrent à des tâches, t_{ij}^k sera la longueur de l'intervalle de temps qui s'écoule entre la fin de la tâche i et le début de la tâche j. Sinon, au cas où le nœud i (resp. j) représente un dépôt, t_{ij}^k sera la durée entre le départ du dépôt k (resp. la fin de la tâche T_i) et le début de la tâche T_j (resp. l'arrivée au dépôt k).

Une paire ordonnée $(T_i,\ T_j)$ est dite compatible si et seulement si le même véhicule peut effectuer les tâches T_i et T_j dans cet ordre tout en respectant la contrainte $b_i + t_{ij} \leq a_j$. Les arcs de A^k peuvent relier deux tâches compatibles ou encore une tâche et un dépôt.
L'ensemble A^k est donc constitué de trois sous-ensembles d'arcs : les arcs émanant du dépôt k en direction de chacune des tâches, les arcs inter-tâches et les arcs partant de chacune des tâches pour rejoindre le dépôt k. En dénotant par I le sous-ensemble des arcs inter-tâches, il est évident que $I \subseteq N \times N$ et l'arc $(i,\ j) \in I$ si la paire de tâches $(T_i,\ T_j)$ est compatible. En définitive, $A^k = I \cup (\{n+k\} \times N) \cup (N \times \{n+k\})$ et le réseau de base sur lequel repose le problème *MDVSP* n'est autre que $G = (V,\ A)$ avec $V = \cup_{k \in K} V^k$ et $A = \cup_{k \in K} A^k$.

Une tournée réalisable est une suite de tâches compatibles deux à deux, couvertes par un même véhicule. Chaque tournée doit commencer et se terminer au même dépôt. Soit c_{ij}^k le coût encouru pour effectuer la tâche T_j immédiatement après la tâche T_i par un véhicule du dépôt D_k. Si un coût est imputé à la tâche T_i, celui-ci est ajouté au coût de l'arc $(i,\ j)$. Notons également par $c_{n+k,j}$ (resp. $c_{i,n+k}$) le coût à supporter si un véhicule stationné au dépôt D_k commence (resp. termine) sa tournée par la tâche T_j (resp. T_i). Aussi, le coût d'une tournée $(T_{i_1},\ T_{i_2}, \ldots, T_{i_h})$ effectuée par un véhicule du dépôt D_k est la somme des coûts des arcs empruntés par le véhicule. Nous remarquerons que les coûts c_{ij}^k des arcs $(i,j) \in A^k$ sont indépendants de k lorsque les arcs $(i,\ j) \in I$. A cet égard, nous nous suffirons de la notation c_{ij} pour ce type de coûts. Par contre, les coûts $c_{n+k,j}$ et $c_{j,n+k}$ dépendent pleinement de k.

Finalement, l'objectif, subordonné à la définition même des coûts, peut être, entre autres :

1. la minimisation du nombre de véhicules à utiliser, auquel cas il faudrait imposer que $c_{n+k,j} = 1$ et $c_{i,n+k} = 0$, $\forall (i,j) \in N \times N$ et $\forall k \in K$ et $c_{ij} = 0$, $\forall (i,\ j) \in I$.
2. la minimisation des coûts d'opération lorsque c_{ij}, $c_{n+k,j}$ et $c_{i,n+k}$ sont les coûts réels de l'utilisation des véhicules (incluant les coûts du véhicule, les coûts de déplacement, le salaire du chauffeur, etc.)
3. la combinaison de 1. et 2.

Remarque : Si tous les coûts $c_{n+k,j}$ et $c_{i,n+k}$ étaient indépendants du dépôt, alors le problème se réduirait au cas à un seul dépôt, connu dans la littérature sous l'appellation (*SDVSP : Single Depot Vehicle Scheduling Problem*) lequel est un problème de flot à coût minimum pouvant être résolu en temps polynomial [9].

1.1.2 Formulation mathématique en un problème multi-flots

Pour la formulation de *MDVSP* en un problème multi-flots ([13], [28]), définissons les variables de flot X_{ij}^k associées aux arcs de G^k pour tout $(i,\ j) \in A^k$ et $k \in K$. X_{ij}^k est une variable binaire qui prend la valeur 1 si et seulement si l'arc $(i,\ j)$ est utilisé dans une tournée qui commence et se termine au dépôt D_k et $X_{ij}^k = 0$ sinon.
Soient $\delta^+(i)$ et $\delta^-(i)$ respectivement l'ensemble des successeurs et des prédécesseurs de i $(i = 1, \ldots, n)$:

$$\delta^+(i) = \{j : (i,j) \in A^k,\ k \in K\}$$
$$\delta^-(i) = \{j : (j,i) \in A^k,\ k \in K\}.$$

Le programme mathématique ainsi associé au problème s'écrit comme suit (Ribeiro et Soumis [28]) :

$$Min \sum_{k \in K} \sum_{(i,j) \in A^k} c_{ij} X_{ij}^k \tag{1.1}$$

$$\sum_{k \in K} \sum_{j \in \delta^+(i)} X_{ij}^k = 1 \qquad i = 1, 2, \ldots, n \tag{1.2}$$

$$\sum_{j \in \delta^-(i)} X_{ji}^k - \sum_{j \in \delta^+(i)} X_{ij}^k = 0 \qquad \forall k \in K, i = 1, 2, \ldots, n, n+k \tag{1.3}$$

$$\sum_{j=1}^{n} X_{n+k,j}^k \leq n_k \qquad \forall k \in K \tag{1.4}$$

$$X_{ij}^k \in \{0, 1\} \qquad \forall (i,j) \in A^k \qquad \forall k \in K. \tag{1.5}$$

L'objectif (1.1) traduit la minimisation du coût d'une affectation des tournées aux véhicules en sommant les coûts de toutes les tournées. Les contraintes (1.2) garantissent que chaque tâche soit couverte exactement une fois. Les contraintes de conservation de flot (1.3) assurent la faisabilité des tournées pour chaque dépôt k. Les contraintes sur le nombre de véhicules disponibles au niveau du dépôt k sont données par les inégalités (1.4). Les contraintes (1.5) sont les contraintes d'intégralité de la solution.

1.1.3 Formulation en un problème de partitionnement

Le problème *MDVSP* peut être formulé en utilisant comme variables de décision les variables de flot associées aux chemins du réseau G défini antérieurement. Chacun de ces chemins constitue une tournée réalisable. Soit Ω l'ensemble des tournées réalisables. Pour $p \in \Omega$, désignons par c_p le coût de la tournée p. Ce coût est fourni par la somme des coûts des arcs qui forment la tournée. En outre, nous définissons les constantes binaires a_{ip} et b_p^k telles que :

$$a_{ip} = \begin{cases} 1 & \text{si et seulement si la tournée } p \text{ couvre la tâche } T_i \\ 0 & \text{sinon,} \end{cases}$$

$$b_p^k = \begin{cases} 1 & \text{si et seulement si la tournée } p \text{ commence et se termine au dépôt } D_k \\ 0 & \text{sinon.} \end{cases}$$

Aussi, en notant θ_p les variables de chemin, le problème *MDVSP* se formule comme suit :

$$Min \sum_{p \in \Omega} c_p \theta_p \tag{1.6}$$

$$\sum_{p \in \Omega} a_{ip} \theta_p = 1 \qquad i = 1, 2, \ldots, n \tag{1.7}$$

$$\sum_{p \in \Omega} b_p^k \theta_p \leq n_k \qquad \forall k \in K \tag{1.8}$$

$$\theta_p \in \{0, 1\} \qquad \forall p \in \Omega. \tag{1.9}$$

La fonction objectif (1.6) minimise le coût total des tournées effectuées. L'ensemble des contraintes (1.7) imposent que chaque tâche i soit exécutée dans exactement une seule tournée par un seul véhicule. Les contraintes (1.8) servent à vérifier que le nombre de tournées utilisant des véhicules du dépôt D_k ne dépasse pas la capacité de ce dépôt. Les contraintes (1.9) sont les contraintes binaires.

Remarque : La formulation (1.6)-(1.9) s'obtient à partir de la formulation (1.1)-(1.5) en appliquant le principe de décomposition de Dantzig-Wolfe. La fonction objectif (1.1) et les contraintes de couverture (1.2) restent au niveau du problème-maître alors que les contraintes (1.3) à (1.4) se retrouvent dans les sous-problèmes.
Bien que la formulation en un problème de partitionnement ait l'avantage de contenir beaucoup moins de contraintes, il n'en demeure pas moins qu'elle fait intervenir un nombre impressionnant de variables, dont l'énumération pose à elle seule un problème de taille.

1.2 Problème *MDVSP* à horizon long

Les problèmes de tournées de véhicules sont généralement traités par rapport à des horizons temporels spécifiés. Un tel horizon est, par exemple, d'une journée régulière pour le cas particulier du transport urbain par autobus. Toutefois, plusieurs situations pratiques engagent des horizons plus longs, allant d'une semaine à un mois ou peut-être plus, tel qu'en transport aérien. L'intérêt de cette section est d'analyser la structure du réseau en cas de prolongement de l'horizon en des multiples d'une période unitaire donnée. A cet effet, nous commencerons par générer des instances de problèmes de tournées de véhicules multi-dépôts pour un horizon variant d'une journée à une semaine, avant d'analyser l'impact de l'horizon sur la structure du réseau.

1.2.1 Génération des problèmes test

Des batteries de problèmes test ont été générés, pour différents horizons. Ces problèmes ont été générés conformément au modèle décrit par Carpaneto *et al.* [6]. Il s'agit notamment de systèmes de transport urbain consistant en un nombre n de tâches, où chaque tâche correspond à un parcours de courte ou de longue durée, lesquelles doivent être accomplies sur un horizon d'une journée régulière.

Pour le cas de notre étude, nous envisageons des horizons variant entre une journée et une semaine. D'abord, il serait utile de noter qu'au lieu de commencer à 5 :00 a.m. et se terminer à 8 :00 p.m. tel le cas d'une journée régulière, une journée complète dure désormais 24 heures. Par conséquent, huit types de problèmes sont considérés, allant d'une journée complète à une semaine de sept jours, en incluant la journée de travail régulière.

Dès lors que l'algorithme de génération d'un problème test pour des tournées d'une journée régulière a déjà été décrit dans des travaux antérieurs (voir, par exemple, [20] et [28]), nous nous limitons ici à son extension à des horizons plus longs.
Chaque journée conserve les mêmes périodes de pointe, à savoir : 7 :00 a.m-8 :00 a.m. et 5 :00 p.m-6 :00 p.m. En outre, nous avons gardé communs à tous les horizons, le nombre de tâches, le nombre de dépôts, le nombre de périodes de pointe par jour, de même que les intervalles sur les durées des tâches. Les seuls paramètres qui se vîrent altérés sont principalement les dates de début et de fin des tâches ainsi que leur dispersion à travers la période considérée.

Ceci nous amène à reconsidérer les pourcentages des tâches de courte durée sur chacune des tranches de temps délimitées par les périodes de pointe. Cependant, les pourcentages doivent rester globalement les mêmes pour chaque type de tâche mais aussi pour chaque type de période (pointe/ hors-pointe). Par exemple, pour un horizon d'une journée complète, il y a deux périodes de pointe et trois périodes normales. Le pourcentage de tâches de courte durée pour chacune des périodes de pointe sera de 15% (totalisant 30%) et 10%, 50% et 10% successivement pour les autres périodes (totalisant 70%). Et c'est cette stratégie qui va permettre d'allonger l'horizon à plus d'une journée. Pour un horizon de 2 jours, il y a un total de quatre périodes de pointe pour lesquelles seront comptées individuellement 7.5% de tâches de courte durée (totalisant 30%) et cinq périodes normales avec, dans l'ordre, 5%, 25%, 10%, 25% et 5% (totalisant 70%).
Pour des horizons de 3 jours et plus, la même technique est adoptée.

1.2.2 Analyse de la structure des réseaux générés

Nous avons d'abord généré des batteries de 10 problèmes pour chaque horizon. En analysant la structure des réseaux générés, pour des instances de problèmes à 500

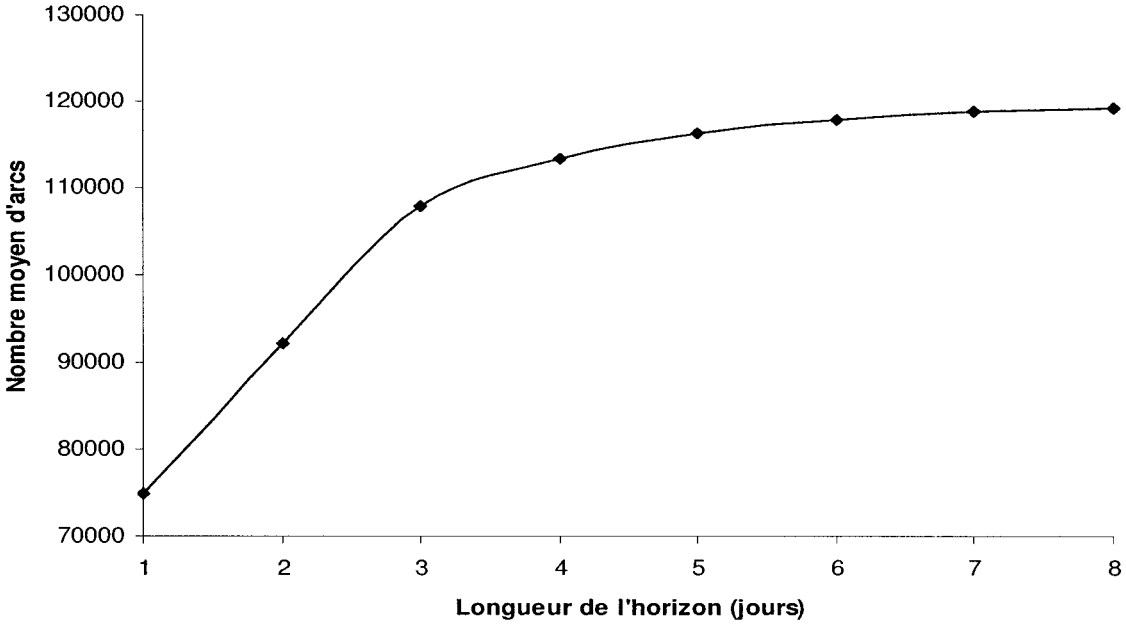

Figure 1.1 – Variation du nombre d'arcs compatibles en fonction de l'horizon

nœuds-tâches et 3 dépôts, nous avons pu dégager une certaine tendance du nombre moyen d'arcs compatibles en fonction de la longueur de l'horizon, telle qu'illustrée par la figure 1.1. Pour une journée régulière, le nombre d'arcs moyen est de 74854.7. Avec des tâches légèrement plus dispersées, le nombre d'arcs moyen pour une journée complète atteint 92200.9, soit une augmentation de près de 23.17%. Cette tendance est plus ou moins similaire pour un horizon de 2 jours pour lequel l'accroissement du nombre d'arcs est de 16.94% en moyenne. Toutefois, dès que l'horizon dépasse 2 jours, les variations du nombre d'arcs deviennent relativement faibles, avoisinant les 5.14% pour un horizon de 3 jours, et n'excédant pas 2.54% pour un horizon de 4 jours. Pour un horizon de 5 jours, le nombre d'arcs additionnel moyen ne dépasse pas 1.37%. Ainsi, il parait évident qu'à mesure que l'horizon s'allonge, le nombre d'arcs augmente avec un taux tendant vers 0%. L'augmentation du nombre d'arcs d'un horizon à l'autre s'explique aisément par l'accroissement de la dispersion des tâches

qui accompagne l'allongement de l'horizon, favorisant plus de compatibilité des arcs. Il est clair aussi que si l'extension de l'horizon est effectuée en un multiple de la journée de base, les tâches seront déplacées à travers le temps en une translation d'une même amplitude et ceci n'affectera aucunement l'ensemble des arcs compatibles.

1.3 Conclusion

Dans ce chapitre, nous avons présenté, à travers des instances de probèmes test le concept d'horizon long dans le cas particulier du problème de tournées de véhicules multi-dépôts. L'augmentation du nombre d'arcs qui accompagne l'extension de l'horizon implique un nombre plus important de variables de décision, quelque soit la formulation considérée. De là, la taille des programmes linéaires à résoudre devient plus grande, d'où la nécessité de faire appel à des techniques de résolution plus appropriées. Logiquement, les méthodes de génération de colonnes se trouvent adaptées à pareilles situations. Le chapitre qui suivra sera consacré à une présentation succinte de ce type d'approches, mettant par la même occasion l'accent sur la problématique posée par le problème de tournées de véhicules multi-dépôts à horizon long.

CHAPITRE 2 : APPROCHE DE RÉSOLUTION STANDARD DE *MDVSP* À HORIZON LONG

L'approche de résolution standard dont il est question dans ce chapitre est la méthode de génération de colonnes. Nous commençons par en donner un aperçu, jetant plus de lumière sur des aspects qui seraient utiles quant à cerner la problématique posée par les problèmes *MDVSP* à horizon long. Ensuite, nous présentons les résultats numériques obtenus en appliquant cette méthode sous sa forme standard. Pour les problèmes dont l'horizon ne dépasse pas 3 jours, les calculs ont été effectués sur des batteries de 10 problèmes test. Pour des horizons supérieurs à 3 jours, nous nous sommes limités à des batteries de 5 problèmes en raison des temps de calcul très grands. Tel que mentionné antérieurement, hormis la différence des horizons, tous les problèmes sont définis par 500 tâches et 3 dépôts, pareille uniformisation nous parraissant nécessaire pour éviter que la variabilité des résultats ne soit imputée à la taille du réseau associé au problème *MDVSP*.

À travers des résultats que nous obtenons, une série d'hypothèses sont émises à l'endroit du problème *MDVSP* à horizon long, lesquelles constitueront la plateforme à une problématique adéquate.
Dans ce chapitre, nous nous abstenons de présenter une revue de littérature en rapport avec la génération de colonnes. Les résultats de base sur le sujet ainsi qu'une liste exhaustive des références peuvent être trouvés dans [19] et [26].

2.1 Principe de la méthode de génération de colonnes

On considère le problème linéaire suivant, appelé problème maître [11],

$$Min \sum_{j \in P} c_j \theta_j \tag{2.1}$$

$$\begin{aligned} \sum_{j \in P} M_j \theta_j &= b \\ \theta_j &\geq 0 \end{aligned} \tag{2.2}$$

où le nombre de colonnes θ_j $(j \in P)$ est soit très grand soit pratiquement impossible à énumérer préalablement. Lors de la résolution de ce problème par l'algorithme du simplexe, l'évaluation des coûts réduits des variables θ_j est très coûteuse en temps de calcul à cause du nombre très élevé de colonnes. Si π est le vecteur des variables duales actuelles, trouver $min(\bar{c}_j = c_j - \pi^T M_j,\ j \in P)$ requiert $O(|P|)$ évaluations de coût réduit. Si l'ensemble des colonnes peut être décrit à l'aide de l'ensemble Ω et la fonction de coût $c(M)$ $(M \in \Omega)$, il suffit alors de résoudre le problème d'optimisation suivant appelé sous-problème ou oracle :

$$\begin{aligned} &Min\ c(M) - \pi^T M \\ &M \in \Omega \end{aligned}$$

L'efficacité de la résolution du sous-problème dépend de la structure de Ω et de la fonction $c(M)$. Si la fonction $c(M)$ est linéaire et Ω est un polyèdre, l'oracle est un programmme linéaire et sa résolution est beaucoup moins coûteuse que l'énumération explicite de toutes les colonnes. Dans certains cas, on peut décrire les colonnes du problème maître à l'aide des points extrêmes d'un certain ensemble convexe S. L'oracle est alors résolu en remplaçant Ω par S. Mais, afin d'assurer que la solution obtenue est un point extrême de S, $c(M)$ doit être concave, ce qui est évidemment vérifié dans le cas où $c(M)$ est linéaire. Afin d'éviter de manipuler toutes les colonnes

(ou dans le cas où il n'est pas possible de connaître toutes les colonnes au préalable), on résout itérativement une suite de programmes linéaires qui ne tiennent compte que d'un sous-ensemble des colonnes. A chaque itération ℓ, un problème MP_ℓ, appelé problème maître restreint est résolu par l'algorithme du simplexe. La recherche d'une colonne à coût réduit négatif se fait en résolvant l'oracle. Si le minimum est non-négatif (en fait égal à zéro à cause de la présence des variables de base), les coûts réduits des variables hors-base sont tous non-négatifs et la solution de MP_ℓ est optimale pour le problème maître. Si le minimum est négatif, la solution du sous-problème est une colonne à coût réduit négatif. Elle est donc ajoutée au problème maître restreint pour obtenir le problème $MP_{\ell+1}$ qui sera résolu à son tour, et ainsi de suite. L'opération consistant à optimiser le problème maître restreint et à résoudre le sous-problème est appelée une *itération de génération de colonnes* [8].

2.1.1 Convergence

Si $|\Omega|$ est fini ou S est choisi de façon qu'il ait un nombre fini de points extrêmes et de rayons extrêmes, le nombre de colonnes susceptibles d'être générées par le sous-problème est fini. Comme toute colonne ajoutée au problème maître aura son coût réduit supérieur ou égal à 0 à l'optimalité du problème maître restreint, chaque colonne sera générée au plus une fois. Ainsi, le processus converge après un nombre fini d'itérations de génération de colonnes vers une solution optimale du problème maître. Dans le cas où le nombre de colonnes que peut générer le sous-problème est infini, la procédure converge aussi, mais le nombre d'itérations requises peut être infini. Des schémas particuliers doivent être développés pour obtenir des solutions quasi-optimales [11].

2.1.2 Particularités pratiques

Certains sous-problèmes permettent d'obtenir une multitude de colonnes. Ceci permet d'enrichir le problème maître dans le but d'accélérer la convergence. Cependant, si le nombre de colonnes devient très grand, le problème maître restreint devient difficile à manipuler. Ainsi, la taille du problème maître restreint est généralement limitée. Ceci ne pose pas de problème de convergence en pratique. En fait, une fois que l'objectif du problème a assez diminué, le nombre de colonnes générées est de plus en plus petit et, après un nombre fini d'itérations, le problème garde toutes ses colonnes. Généralement, la résolution du problème maître est une sous-routine d'un processus récursif de séparation et d'évaluation. À chaque nœud de l'arbre de branchement, le problème maître est résolu par génération de colonnes. Dans certains cas, il peut être possible de n'utiliser que des bornes inférieures et supérieures fournies par cette résolution [11].

2.2 Application au *MDVSP*

La méthode de résolution par génération de colonnes a été appliquée à la relaxation linéaire du problème *MDVSP*. Tous les calculs ont été effectués à l'aide du logiciel *GENCOL* sur un serveur d'exécution Origine 2000 de SUN Microsystem [Microprocesseurs $400Mhz$].

Pour l'interprétation des résultats produits par *GENCOL*, nous avons considéré le nombre d'arcs par réseau (**ArcRes**), le nombre d'itérations de génération de colonnes (**ItGen**), le nombre de problèmes de plus court chemin résolus (**NbPCC**), le nombre de colonnes générées (**Col**) et le temps CPU total (**TL-cpu**). Les résultats obtenus sont rassemblés dans les tableaux A.1 et A.2 de l'annexe A. Au passage, nous soulignons que les résultats ayant trait aux problèmes des horizons de 7 jours n'ont pu

être obtenus quoi que le processus de calcul ait duré pas moins de 5 mois avant d'être interrompu. Les problèmes portant sur cet horizon ont été écartés pour cette phase de l'étude en raison du coût exorbitant de leur résolution en termes de temps CPU. Pour l'horizon de 6 jours, deux problèmes seulement ont pu être résolus.

Tableau 2.1 – Résultats moyens de la méthode de génération de colonnes

Horizon	**ArcRes**	**ItGen**	**NbPCC**	**Col**	**TL-cpu** (s)
Journée régulière	74854.7	172.5	514.5	11639.6	368.3
Journée complète	92200.9	385.7	1154.1	19249.6	1267.0
2 jours	107821.0	1070.9	3209.7	35776.2	4215.0
3 jours	113369.6	3631.6	10891.6	95621.2	32437.5
4 jours	116256.4	155791.2	23514.2	186779.8	155791.2
5 jours	117845.2	22229.4	66685.2	497349.4	515726.9
6 jours	119891.5	14333.5	42997.5	333563.5	682632.5

Sur la base des résultats obtenus, dont les moyennes sont présentées dans le tableau 2.1, il y a lieu de constater certains comportements, en moyenne, des problèmes résolus, observés du point de vue de la variation de l'horizon seul. Aussi, les points ci-dessous sont à relever :

Augmentation de la taille des réseaux : À première vue, les résultats obtenus, la figure 2.1 à l'appui, permettent d'observer ce qui suit :

- Dans le cas des problèmes se rapportant à une journée régulière, le temps CPU moyen est de 368.3 s, se multipliant par un facteur de 3.44 dès que l'on passe à une journée complète, pour tripler une autre fois pour un horizon de deux jours.
- Les temps de calcul évoluent de manière exponentielle de sorte que, à partir d'un horizon de seulement 3 jours, une augmentation de 5.14% du nombre d'arcs moyen multiplie le temps CPU par un facteur de 7.69. Ce facteur multiplicatif garde sensiblement le même ordre de grandeur pour un horizon de 4 jours pour lequel le nombre d'arcs n'augmente en moyenne que de 2.54%. Pour un horizon de 5 jours,

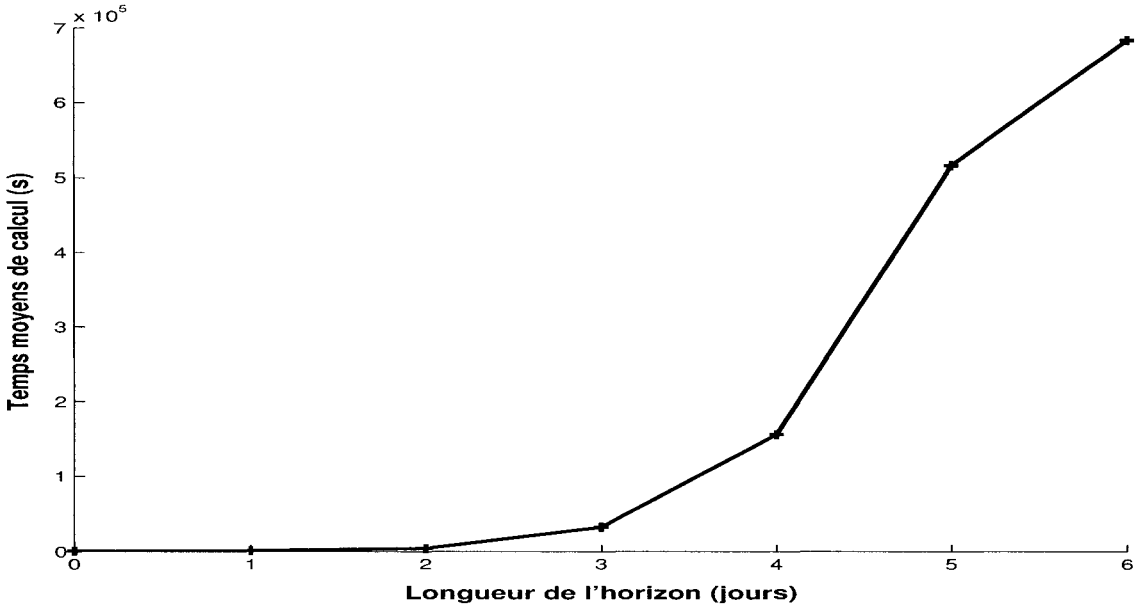

Figure 2.1 – Variation du temps CPU moyen en fonction de l'horizon

le temps CPU moyen triple, atteignant 143 heures, au moment où le nombre moyen d'arcs additionnels n'a pas excédé la proportion de 1.37%.

À la lumière de ce qui a précédé, une interprétation "hâtive" de la variabilité des temps CPU veut que le réseau se trouve plus dégagé lorsque l'horizon est plus étendu, augmentant ainsi les intervalles de temps inter-tâches et, de là, le nombre d'arcs compatibles et la taille du problème à résoudre. Cette hypothèse se verra rejetée plus loin, à mesure que seront divulgués d'autres détails, encore plus pertinents.
À cet égard, nous nous proposons d'examiner de plus près les fichiers résultats produits par *GENCOL* pour tenter de discerner d'autres aspects susceptibles d'expliquer l'augmentation de la difficulté des problèmes *MDVSP* à horizon long notamment, ceux liés au processus de génération de colonnes.

Dégénérescence du problème primal : La figure 2.2 représente la valeur de l'objectif en fonction du nombre d'itérations de génération de colonnes, obtenue pour les

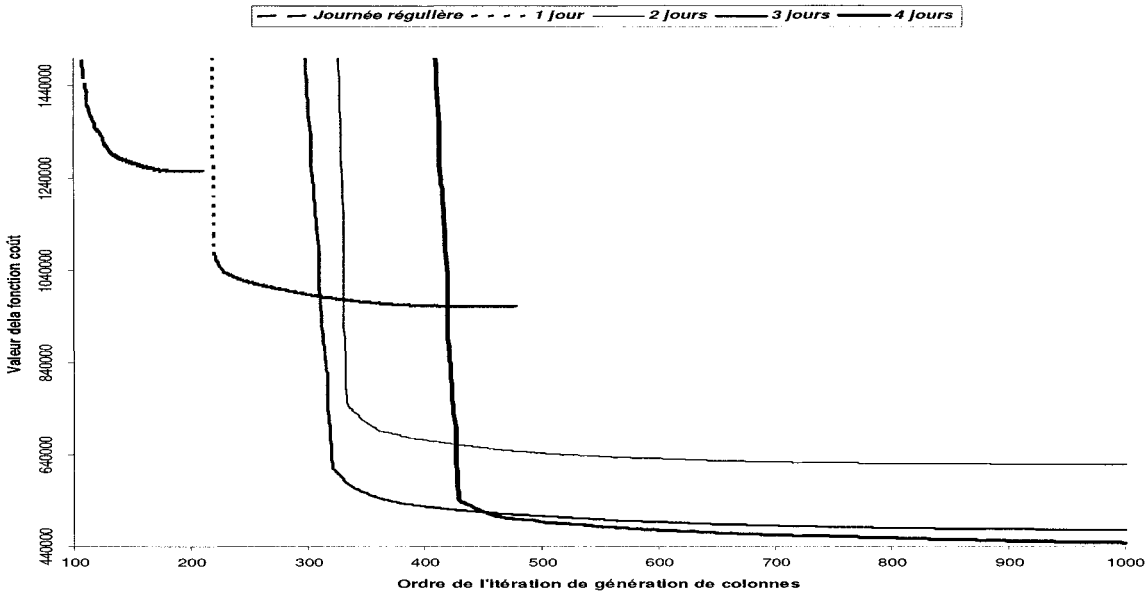

Figure 2.2 – Effet de queue de la fonction coût

cas des problèmes les plus difficiles se rapportant à chacun des horizons considérés. Au regard de la variation de la valeur de la fonction coût, certaines remarques s'imposent :

- Qu'importe l'horizon considéré, le taux de variation de la valeur de l'objectif en fonction du nombre d'itérations de génération de colonnes est pratiquement le même antérieurement à un certain point à partir duquel le taux de variation diminue brusquement et commence à s'approcher de zéro. Plus l'horizon est étendu, plus éloigné se retrouve ce point de la courbe.
- L'aplatissement de la courbe est d'autant plus prononcé que l'horizon augmente, présentant des planchers qui s'étendent sur plusieurs milliers d'itérations.

En fait, l'une des caractéristiques principales du comportement de l'algorithme de génération de colonnes est la lenteur de la convergence à la fin de l'algorithme. La figure 2.2 montre un comportement typique de l'algorithme. On y voit clairement l'effet de queue pour les dernières itérations. Le nombre d'itérations nécessaires pour

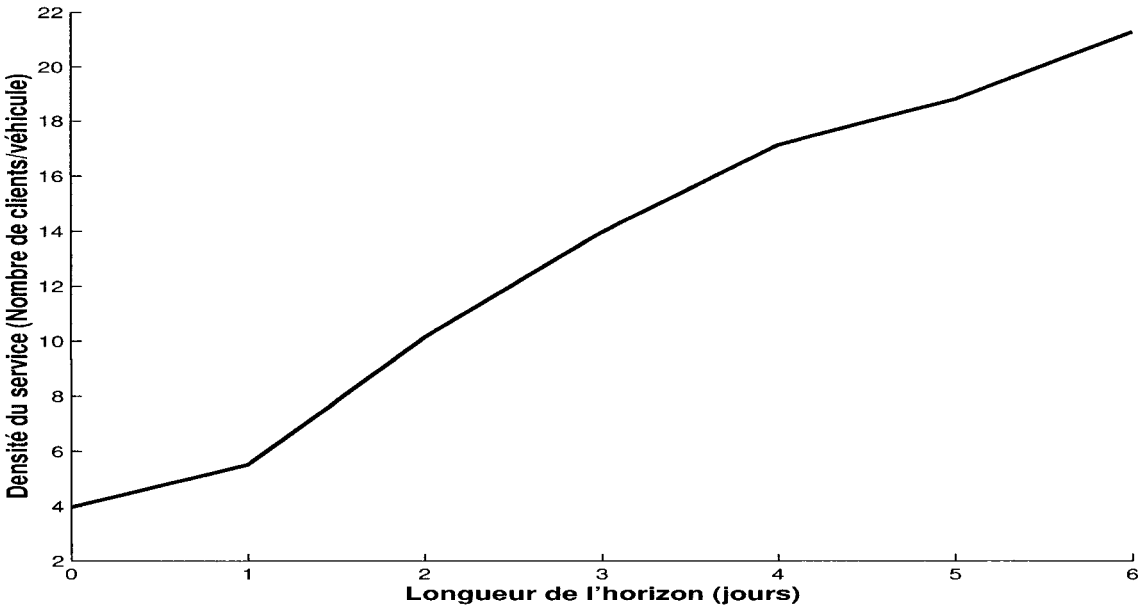

Figure 2.3 – Variation de la densité du service

une amélioration minime de l'objectif est très élevé. Ceci est dû essentiellement à la dégénérescence du problème primal.

Étirement des chemins : D'un autre côté, la variation de la densité du service (figure 2.3) laisse entrevoir que les chemins deviennent plus longs à mesure que l'horizon s'étend (le nombre de parcours effectués augmente), rendant encore plus lent le processus de génération des colonnes en augmentant le nombre de 1 par colonne. Ce comportement apparaît de façon plus intéressante sur la courbe de variation du temps CPU moyen en fonction de la densité du service (figure 2.4).

2.3 Conclusion

Le long du deuxième chapitre, nous sommes parvenus à résoudre, par la méthode de génération de colonnes standard, des problèmes de tournées de véhicules multi-

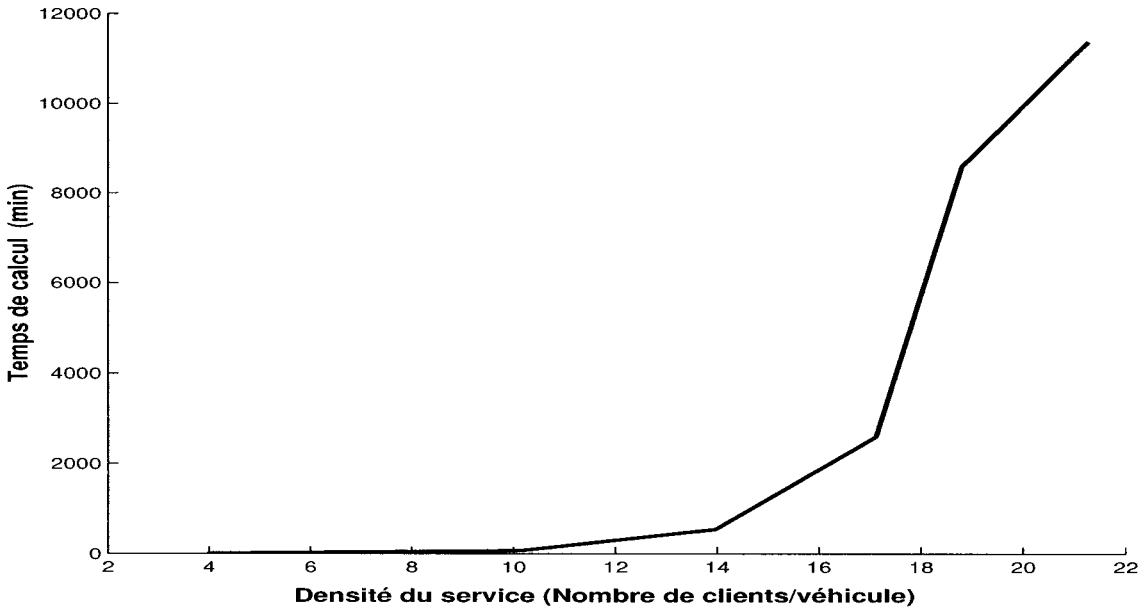

Figure 2.4 – Variation du temps CPU moyen en fonction de la densité du service

dépôts, jusqu'à un horizon temporel au niveau duquel la résolution s'est avérée pratiquement impossible. Tenant compte de maints paramètres techniques, il a été plus ou moins possible de cerner la problématique des problèmes *MDVSP* à horizon long. Une série de facteurs susceptibles de constituer la source de cette problématique furent explicitement énumérés sans pour autant trancher en faveur de l'un ou l'autre. Néanmoins, cette problématique renvoie expressément aux difficultés de convergence de l'algorithme de génération de colonnes, dues essentiellement à la dégénérescence primale, à la densité des colonnes générées , à l'oscillation des valeurs duales durant le processus de génération de colonnes. À quel degré intervient cet aspect pour le cas particulier du problème *MDVSP* à horizon long ? Y a-t-il une quelconque issue pour y pallier ? Autant de questions auxquelles nous tenterons d'apporter des éléments de réponse convaincants dans les prochaines sections.

CHAPITRE 3 : PROCÉDURE DE RÉDUCTION DU RÉSEAU

Au deuxième chapitre, nous avons montré que la taille des réseaux associés aux problèmes de tournées de véhicules à horizon long augmente à mesure que l'horizon temporel s'étend. Par ailleurs, ceci s'accompagnant d'un accroissement significatif de la difficulté des problèmes, il nous a été offert d'émettre une hypothèse imputant, a priori, l'aspect "difficulté" à la taille des problèmes. Dans les sections qui viennent, notre objectif est de vérifier la véracité d'une telle assertion. Il s'agira, en l'occurrence, d'opérer une réduction du réseau, conformément au principe de *fixation des variables* ([3], [13]). Aussi, l'approche suggérée opère en trois étapes, à savoir :

- Calcul d'une borne inférieure sur la valeur optimale de la relaxation linéaire du problème *MDVSP*.
- Calcul d'une borne supérieure sur la valeur optimale du problème en nombres entiers.
- Réduction de la taille du problème *MDVSP* et résolution de la version réduite par la méthode de génération de colonnes standard.

Nous commençons par développer chacun de ces points en en citant les idées de base ainsi que les principaux résultats auxiliaires qui en découlent, lesquels résultats sont également utiles pour des applications autres que celle en cours. Par la suite, nous détaillons l'algorithme de réduction du réseau, de même que les extensions qui y sont afférentes, situant la contribution majeure de la présente étude dans le contexte des travaux de recherche antérieurs. De là, nous procédons à une application de l'algorithme de réduction du réseau sur des problèmes qui sont choisis parmi ceux générés au deuxième chapitre. La réduction du réseau est entreprise préalablement à la résolution de chacun de ces problèmes par la méthode de génération de colonnes.

3.1 Principe de l'approche

La réduction de la taille du problème *MDVSP* en termes de nombre de variables de flot est l'idée de base de l'approche développée. Cette idée repose essentiellement sur le principe de la *fixation de variables* proposé par Bianco *et al.* [3] (Proposition 3.1, p. 5). En outre, la borne inférieure ainsi que la solution primale entière correspondante, nécessaires à la procédure de réduction, sont calculées en utilisant l'heuristique décrite par Ben Amor [1]. L'opération de réduction consiste en la suppression systématique d'autant d'arcs que possible au niveau du réseau G défini au premier chapitre. Les arcs ciblés sont les arcs *improductifs*.

Définition : Un arc est dit *improductif* si son insertion dans une solution réalisable n'améliore en rien la valeur de la fonction coût. En d'autres termes, la contribution d'un tel arc dans la composition des tournées de la solution optimale est sans aucune valeur.
Dans le cas de la présente approche, nous adoptons les coûts réduits comme critère d'élimination des arcs *improductifs*, laquelle idée sera explicitée plus en détail ultérieurement [13].

3.1.1 Calcul d'une borne inférieure

À ce niveau, nous résolvons le problème réseau associé au *MDVSP* de base. A cet effet, tous les dépôts D_k $(k = 1, \ldots, |K|)$ sont agrégés en un dépôt *fictif* unique D où sont regroupés tous les véhicules. Ce dépôt sera représenté dans le réseau $G = (V, A)$ par le nœud 0. N désignant l'ensemble des nœuds-tâches du réseau G et I l'ensemble des arcs compatibles alors $A = I \cup (\{0\} \times N) \cup (N \times \{0\})$ et $V = N \cup \{0\}$.
Au moment où les coûts inter-tâches demeurent inchangés, les coûts des arcs $(0, j)$

et $(i, 0)$ sont calculés de sorte que :

$$c_{0,j} = \min_{k=1,\ldots,|K|} c_{k+n,j} \qquad \forall j = 1, \ldots, n \tag{3.1}$$

$$c_{i,0} = \min_{k=1,\ldots,|K|} c_{i,k+n} \qquad \forall i = 1, \ldots, n. \tag{3.2}$$

Le problème ainsi défini s'apparente au problème de tournées de véhicules à un seul dépôt, communément nommé *SDVSP*. Tel que mentionné au premier chapitre, ce problème est une relaxation de *MDVSP*. Conséquemment, un modèle mathématique approprié pour décrire ce problème peut prendre la forme que voici :

$$Min \sum_{(i,j)\in A} c_{ij} X_{ij} \tag{3.3}$$

$$\sum_{j\in\delta^+(i)} X_{ij} = 1 \qquad i = 1, 2, \ldots, n \tag{3.4}$$

$$\sum_{j\in\delta^-(i)} X_{ji} - \sum_{j\in\delta^+(i)} X_{ij} = 0 \qquad i = 0, \ldots, n \tag{3.5}$$

$$\sum_{j=1}^{n} X_{0,j} \leq n_v \tag{3.6}$$

$$X_{ij} \in \{0, 1\} \qquad \forall (i, j) \in A \tag{3.7}$$

où n_v est le nombre de véhicules disponibles :

$$n_v = \sum_{k=1}^{|K|} n_k.$$

Alternativement, la contrainte de conservation de flots peut être substituée par :

$$\sum_{j\in\delta^-(i)} X_{ji} = 1 \qquad i = 1, 2, \ldots, n \tag{3.8}$$

Le problème ainsi formulé se résout aisément en utilisant le logiciel *Cplex 6.6* [15]. Pratiquement, nous avons pu vérifier que la formulation contenant la contrainte (3.8)

se résout en un temps CPU nettement meilleur par opposition à la première, quelque soit la procédure adoptée (*opt* ou *netopt*) mais il n'en demeure pas moins que ce temps CPU est de l'ordre de quelques secondes. Nous noterons Z_s la valeur optimale du *SDVSP*.

3.1.2 Calcul d'une borne supérieure

Une fois résolu, le problème *SDVSP* fournit les tournées optimales par rapport au dépôt *fictif*. Soit n_τ le nombre de ces tournées. Ignorant le dépôt *réel* auquel correspond chacune des tournées, nous pouvons isoler n_τ chemins de coût minimum que nous noterons CCM_t dont les coûts respectifs sont :

$$C_t = \sum_{(i,j) \in CCM_t} c_{ij} \qquad t = 1, \ldots, n_\tau.$$

En raccordant chacun des chemins CCM_t à chaque dépôt D_k du réseau initial, nous produisons $|K| \times n_\tau$ tournées réalisables pour le problème *MDVSP*. Restreignant la composition de l'ensemble Ω aux tournées ainsi construites, nous résolvons la formulation mathématique en un problème de partitionnement (1.6)-(1.9) avec c_{tk} les coûts associés à chacune des colonnes :

$$c_{tk} = C_t + c_{n+k,d_t} + c_{f_t,n+k} \qquad t = 1, \ldots, n_\tau \qquad k = 1, \ldots, |K|$$

où d_t et f_t sont, respectivement, le nœud de début et le nœud de fin du chemin CCM_t $(t = 1, \ldots, n_\tau)$.

Le problème ainsi formulé comprend $|K| \times n_\tau$ variables et $n + |K|$ contraintes. L'utilisation du logiciel *Cplex 6.6* [15] fournit la solution optimale entière en des temps très faibles (de l'ordre de la fraction de seconde). Notons par $\bar{Z}$ la valeur de cet optimum. Pour l'obtention de $\bar{Z}$, la résolution du problème (1.6)-(1.9) étant menée sur un domaine restreint de Ω, $\bar{Z}$ est une borne supérieure sur la valeur optimale entière de *MDVSP*.

3.2 Algorithme de réduction du réseau *MDVSP*

De prime abord, en désignant par Z_m la valeur optimale de la relaxation linéaire du problème *MDVSP*, la double inégalité suivante est toujours vérifiée :

$$Z_s \leq Z_m \leq \bar{Z}. \tag{3.9}$$

Revenant à la section 3.1.1, la résolution de *SDVSP* permet également d'obtenir les coûts réduits $\bar{c}_{ij}$ associés aux variables de flot X_{ij} à l'optimalité pour tout arc $(i,j) \in A$. Aussi, il est important de souligner que l'introduction de toute variable de flot à coût réduit positif dans la solution optimale du *SDVSP* résulterait en une augmentation de la valeur Z_s de l'optimum, ce qui est a priori indésirable pour le cas du *SDVSP*. Néanmoins, une telle propriété serait plus intéressante vue d'un autre angle. Par référence à (3.9), une augmentation de la valeur de la borne inférieure Z_s permettrait d'approcher Z_m tout en se gardant de ne pas dépasser la borne supérieure $\bar{Z}$. Pareil critère peut être exploité efficacement pour la sélection des arcs candidats à former la solution optimale entière de *MDVSP*. Calculons l'écart $E = \bar{Z} - Z_s$.
Connaissant les coûts réduits $\bar{c}_{ij}$, tous les arcs $(i,j) \in A$ candidats à former la solution optimale du *MDVSP* doivent au moins vérifier l'inégalité suivante :

$$\bar{c}_{ij} \leq E. \tag{3.10}$$

Sur cette base, tous les arcs $(i,j) \in A$ qui ne satisfont pas la condition (3.10) formeront l'ensemble des arcs *improductifs* lesquels seront définitivement supprimés du réseau $G = (V, A)$.
La démarche ainsi déployée peut être résumée par l'algorithme $\mathcal{A}\ell$:

Étape 0 : *Identification des dépôts Source et Puits*
Les dépôts *Source* et *Puits* sont les dépôts dont les coûts vérifient les équations (3.1) et (3.2). Il sera noté qu'il ne s'agit pas nécessairement d'un même dépôt.

Étape 1 : *Construction du modèle réseau*

Moyennant les coûts c_{ij}, $(i,j) \in A$, ainsi que le nombre de véhicules n_ν, le modèle réseau est défini par le programme linéaire (3.3)-(3.7).

Étape 2 : *Résolution du modèle réseau*

Résoudre le programme linéaire (3.3)-(3.7) et en déduire :

1. Le vecteur des valeurs optimales des variables de flot X_s et l'optimum correspondant Z_s avec $X_s = [X_{ij}]_{(i,j)\in A}$ $(X_{ij} \in \{0,1\})$.
2. Les coûts réduits $\bar{c}_{ij}$ associés aux variables de flots X_{ij} $\forall$ $(i,j) \in A$.
3. Le vecteur $\hat{\pi}_s$, solution duale optimale du problème *SDVSP* associé au réseau G.

L'intérêt du dernier point sera expliqué aux prochains chapitres.

Étape 3 : *Résolution du problème de partitionnement*

Cette étape sera menée conformément aux points suivants :

1. Identifier les n_τ chemins de coût minimum décrits par X_s.
2. Associer chacun des n_τ chemins aux $|K|$ dépôts et calculer les coûts c_{tk} correspondants $(t = 1, \ldots, n_\tau$ et $k = 1, \ldots, |K|)$.
3. Résoudre le problème de partitionnement en nombres entiers et en déduire la valeur de $\bar{Z}$.

Le problème de partitionnement aura pour colonnes les tournées obtenues au point 2 de l'étape 3.

Étape 4 : *Réduction du réseau initial G*

On définit par A_r l'ensemble des arcs du réseau réduit.

1. Calculer l'écart $E = \bar{Z} - Z_s$.
2. Pour tout arc $(i,j) \in A$, comparer son coût réduit $\bar{c}_{ij}$ à E

Si $\bar{c}_{ij} \leq E$ alors $(i,j) \in A_r$

Sinon l'arc (i,j) est *improductif* et il est supprimé.

Notons par $G_r = (V, A_r)$ le nouveau réseau obtenu suite à la réduction ainsi opérée.

3.3 Extension de l'algorithme $\mathcal{A}\ell$ au réseau G_r

L'application de l'algorithme $\mathcal{A}\ell$ permet d'estimer, à l'étape 3, une valeur de la borne supérieure $\bar{Z}$ qui n'est, certes, pas la seule possible. Cette valeur dépend essentiellement de la composition des colonnes mises en jeu dans la résolution du problème de partitionnement. Aussi, une amélioration de $\bar{Z}$ serait possible si de nouvelles colonnes étaient ajoutées aux $|K| \times n_\tau$ colonnes déjà existantes. À cet effet, nous avons pensé à une extension de la procédure décrite plus haut en l'appliquant à nouveau au réseau réduit G_r .

L'objectif principal de l'extension de l'algorithme de réduction du réseau *MDVSP* est de réduire davantage G_r en éliminant d'autres arcs. En voici les principales étapes :

Étape 0' : *Résolution du modèle réseau réduit*

À cette étape, il est question de répéter les étapes 0 à 2 de l'algorithme $\mathcal{A}\ell$ en substituant le réseau réduit G_r au réseau initial G.

Les dépôts *Source* et *Puits* sont à nouveau identifiés avant que le modèle réseau ne soit construit en utilisant comme coûts c_{ij}, $(i,j) \in A_r$. La résolution du programme linéaire (3.3)-(3.7) avec ces nouveaux paramètres donne lieu aux résultats suivants :

1. La solution réseau optimale associée au réseau réduit G_r, que l'on désignera par X_s^r telle que : $X_s^r = [X_{ij}]_{(i,j)\in A_r}$ avec $X_{ij} \in \{0,1\}$.
2. $\hat{\pi}_{sr}$, solution duale optimale du problème *SDVSP* associé au réseau réduit G_r.

Remarque : La valeur de l'optimum Z_s, de même que le nombre n_τ de véhicules mobilisés à l'optimalité, demeurent inchangés, que le problème *SDVSP* soit associé au réseau initial G ou bien au réseau réduit G_r.

Étape 1' : *Résolution du problème de partitionnement étendu*

La principale nouveauté introduite à ce stade consiste à regrouper les colonnes non-identiques de G et G_r pour la construction du problème de partitionnement. Il s'agit, en l'occurrence, d'appliquer la procédure ci-dessous :

1. Identifier les n_τ chemins de coût minimum décrits par X_s^r que l'on désignera par $CCM_{t'}$ $(t' = 1, \ldots, n_\tau)$.
2. Trier les chemins ainsi identifiés de sorte à n'en retenir que les n_τ^r chemins $(n_\tau^r \leq n_\tau)$ qui n'ont pas été recensés lors de l'application de l'algorithme $\mathcal{A}\ell$ de réduction du réseau initial. Autrement, ces chemins sont tels que :
   $CCM_{t'} \cap CCM_t = \emptyset \qquad \forall\, t' = 1, \ldots, n_\tau^r$ et $\forall\, t = 1, \ldots, n_\tau$.
   Les $n_\tau + n_\tau^r$ chemins ainsi regroupés forment la base pour la construction des colonnes du problème de partitionnement.
3. Raccorder chacun des $n_\tau + n_\tau^r$ chemins aux $|K|$ dépôts et calculer les coûts c_{tk} correspondants $(t = 1, \ldots, n_\tau + n_\tau^r$ et $k = 1, \ldots, |K|)$ et construire le problème de partitionnement étendu dont le nombre de colonnes sera $(n_\tau + n_\tau^r) \times |K|$.

La nouvelle valeur de la borne supérieure ainsi obtenue sera :

$$\bar{Z}_r \leq \bar{Z}$$

Étape 2' : *Réduction du réseau réduit G_r*

La réduction du réseau G_r, ou encore la *double réduction* du réseau G, sera menée en conformité avec l'étape 4 de l'algorithme $\mathcal{A}\ell$, avec $\bar{Z}_r$ comme nouvelle borne supérieure. Par conséquent, si $\bar{Z}_r < \bar{Z}$, une *double réduction* est possible. En désignant par $G'_r = (V, A'_r)$ le réseau doublement réduit, nous procèderons comme suit :

1. Calculer le nouvel écart $E_r = \bar{Z}_r - Z_s$.
2. Pour tout arc $(i,j) \in A_r$, comparer son coût réduit$\bar{c}_{ij}$ à E_r

   Si $\bar{c}_{ij} \leq E_r$ alors $(i,j) \in A'_r$

   Sinon $A'_r = A_r \setminus \{(i,j)\}$

Remarque : L'algorithme de réduction peut être exécuté plus de deux fois et les valeurs de $\bar{Z}$ qui sont fournies peuvent être plus petites. Toutefois, avec les problèmes test dont nous disposions, les nouvelles valeurs de $\bar{Z}$ sont telles que l'écart E reste toujours trop grand pour plus de réduction du réseau.

3.4 Résolution des *MDVSP* réduits

Pour resituer le contexte de notre étude, rappelons qu'il s'agit de la résolution de problèmes de tournées de véhicules à horizon long, caractérisés par l'augmentation vertigineuse de la difficulté des problèmes à mesure que l'horizon s'allonge.
Pour des horizons allant de *1 jour à 7 jours*, nous avons résolu par la méthode de génération de colonnes standard des batteries de plusieurs problèmes test. Pour chacun des horizons, nous supposons que le problème le plus difficile est le meilleur représentant de la classe des problèmes de l'horizon. Toutefois, pour mieux corroborer nos conclusions sur l'horizon, nous adjoignons le problème le plus facile. Dans ce qui suivra, nous conviendrons que le numéro impair $(2i+1)$ et le numéro pair $(2i)$ désignent respectivement le problème le plus difficile et le problème le plus facile de l'horizon i ($i = 0, \ldots, 7$ *jours*).

Les seize problèmes réduits ainsi choisis sont résolus par génération de colonnes. Les résultats obtenus sont présentés dans les tableaux 3.1 à 3.2. Pour les notations, Z_s est

la valeur optimale de *SDVSP*, $\bar{Z}$ est la valeur optimale du problème de partitionnement, **ArcIn** est le nombre d'arcs du réseau initial, **ArcRe** est le nombre d'arcs du réseau réduit, **TRed** est le taux de réduction du réseau (arcs supprimés/arcs du réseau initial), **SP-cpu** est le temps CPU requis pour la résolution des sous-problèmes (s), **MP-cpu** est le temps CPU pour la résolution des problèmes maîtres (s) et **TL-cpu** est le temps CPU total (s). Les nombres d'arcs **ArcIn** et **ArcRe** présentés dans le tableau 3.1 se rapportent à chaque sous-réseau k ($k = 1, \ldots, |K|$), signifiant que les nombres totaux d'arcs seraient respectivement **ArcIn** $\times$ $|K|$ et **ArcRe** $\times$ $|K|$ pour chacun des cas considérés.
Ces mêmes problèmes seront retenus pour toute la suite de l'étude.

3.4.1 Réduction de la taille du réseau

La figure 3.1 montre clairement que la procédure de réduction du réseau mise en œuvre est efficace, le taux de réduction dépassant les 80% pour un problème. De plus, la réduction du réseau initial devient d'autant plus significative que l'horizon s'allonge, à tel point que, pour les problèmes des horizons de 4 à 7 jours, les réseaux sont réduits de plus de deux tiers. Pareil résultat est très encourageant dans la mesure où il nous permet de mieux mettre en exergue l'effet, s'il y en a, de la taille du réseau sur le temps de déroulement du processus de résolution. À cet égard, la résolution des problèmes réduits est effectuée par la méthode de génération de colonnes standard sur le logiciel *GENCOL*, avec les mêmes paramètres opérationnels que ceux fixés lors de la résolution des versions initiales (non réduites) de ces problèmes. Les résultats sont présentés au tableau 3.1.

Remarque : Il est à noter qu'au fur et à mesure que l'horizon s'allonge, l'écart entre les valeurs optimales des problèmes *MDVSP* et *SDVSP* diminue (figure 3.2). Cette propriété sera exploitée aux prochains chapitres lorsqu'il y aura lieu de choisir une bonne estimation de la solution duale initiale du problème *MDVSP*.

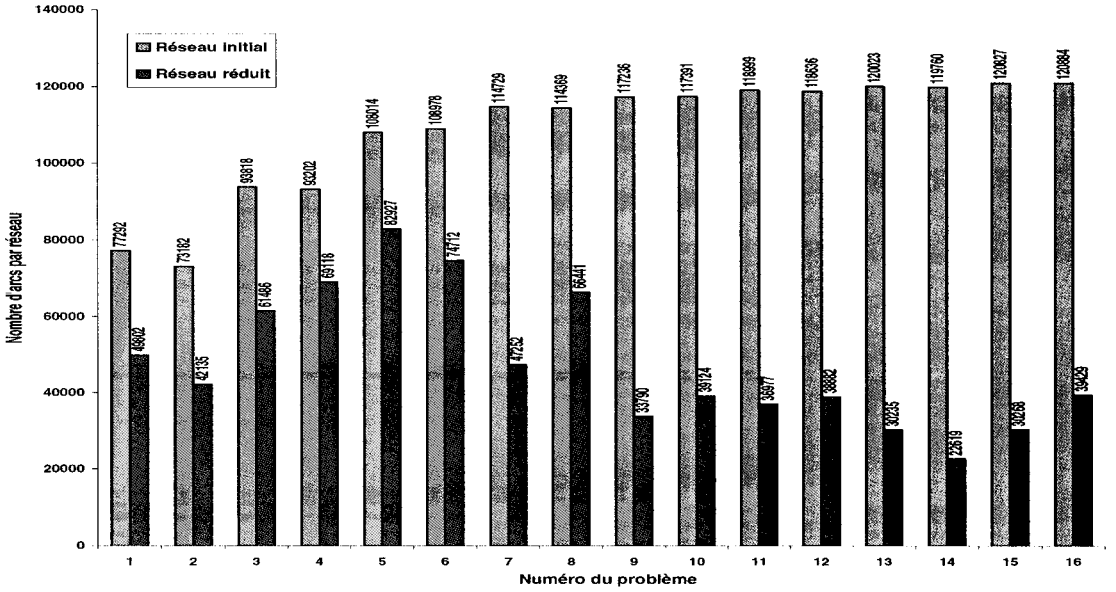

Figure 3.1 – Mesure de l'ampleur de la réduction du réseau

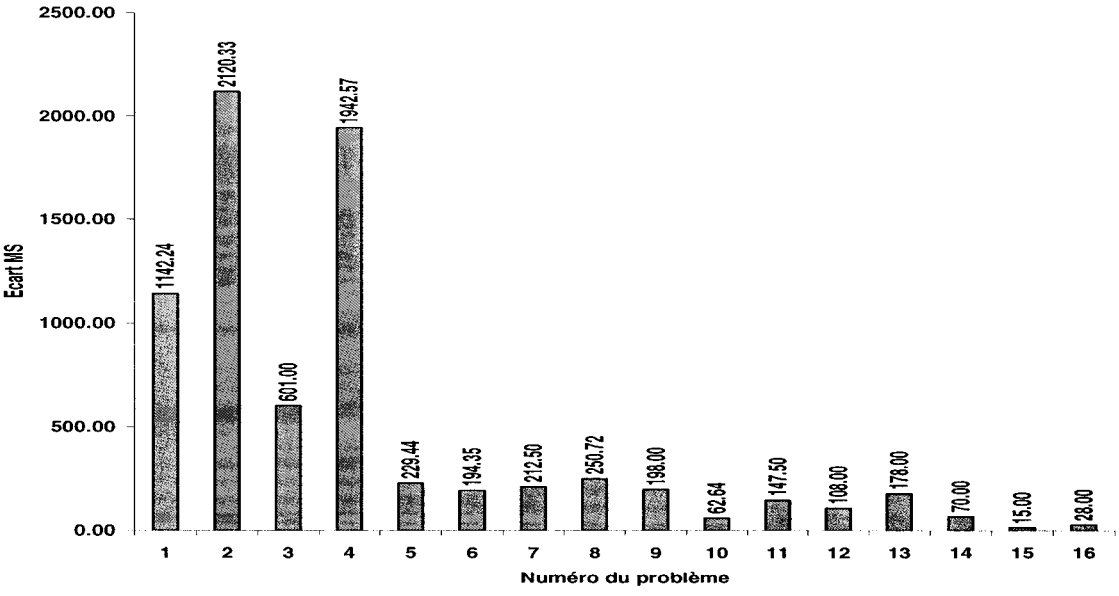

Figure 3.2 – Variation de l'écart entre les valeurs optimales des problèmes *MDVSP* et *SDVSP*

Tableau 3.1 – Effets de la procédure de réduction du réseau

Pb.	Z_s	$\bar{Z}$	E_r	ArcIn	ArcRe	TRed (%)
1	1253697	1259075	5378	77292	49802	35,57
2	1491588	1499549	7961	73182	42135	42,42
3	961026	966234	5208	93818	61486	34,46
4	1081809	1091223	9414	93202	69118	25,84
5	619126	622812	3686	108014	82927	23,23
6	615529	619045	3516	108978	74712	31,44
7	474598	476604	2006	114729	47252	58,81
8	580704	584447	3743	114369	66441	41,91
9	436274	438284	2010	117236	33790	71,18
10	464337	465310	973	117391	39124	66,67
11	458439	460552	2113	118999	36977	68,93
12	472332	475206	2874	118636	38882	67,23
13	401864	403437	1573	120023	30235	74,81
14	454116	455519	1403	119760	22619	81,11
15	440135	440695	560	120827	30268	74,95
16	441142	442060	918	120884	39429	67,38

3.4.2 Effet de la réduction du réseau sur les temps de calcul

Le tableau 3.2 présente les résultats qui ont pu être obtenus en résolvant les versions initiales et réduites de chacun des problèmes 1 à 14. Les problèmes 15 et 16 sur un horizon 7 jours sont restés insolubles même avec une réduction du réseau qui dépasse 2/3. Quoiqu'ayant considérablement contribué à la réduction du réseau, cet approche ne semble cependant pas fournir les résultats escomptés sur le plan des temps de calcul. En effet, la figure 3.3 montre que l'amélioration des temps CPU est généralement négligeable, sauf pour quelques problèmes, malgré de fortes réductions du réseau. Les problèmes de l'horizon 7 jours (problèmes 15 et 16) sont réduits à plus de deux tiers mais restent quand mêmes insolubles par la méthode de génération de colonnes standard. Autant d'éléments qui nous poussent à investiguer de nouvelles

Tableau 3.2 – Résultats *GENCOL* AVANT et APRÈS réduction du réseau

Pb.	AVANT la réduction			APRÈS la réduction		
	MP-cpu	SP-cpu	TL-cpu	MP-cpu	SP-cpu	TL-cpu
1	330,7	167,9	502,6	328,4	74,1	405,0
2	113,8	73,3	190,6	127,2	29,8	160,6
3	1047,7	485,6	1539,4	1141,6	249,1	1394,8
4	462,5	248,6	716,0	468,2	148,2	620,2
5	6391,8	1319,5	7721,2	6632,1	875,1	7515,8
6	4800,3	984,0	5792,8	5766,1	814,5	6591,2
7	71145,9	9092,3	80266,8	27882,2	1157,3	29055,0
8	12140,7	2275,8	14435,5	12752,8	1148,7	13911,6
9	177880,8	10502,2	188419,6	178089,9	1490,7	179608,9
10	129774,4	9112,6	138916,1	62604,7	1135,9	63758,9
11	307053,3	2610,2	309701,3	300545,0	3012,1	303600,7
12	180199,9	12401,2	192641,3	61909,2	1061,2	62990,3
13	902369,5	26324,5	928771,3	587715,5	2275,1	590052,0
14	420377,0	16065,5	436493,7	364970,8	2972,8	367983,1

alternatives pour résoudre le problème *MDVSP* à horizon long en des temps plus raisonnables.

3.5 Conclusion

La procédure développée dans ce chapitre ne semble pas améliorer, du moins à ce stade, les performances de la procédure de génération de colonnes pour la résolution des problèmes de tournées de véhicules multi-dépôts (*MDVSP*) à horizon long. Néanmoins, certains résultats qui ont émergé du processus de réduction du réseau pourraient faire l'objet d'une exploitation dans de futures étapes de l'étude, tel est le cas des valeurs des variables duales des problèmes *SDVSP* dont l'utilité se fera certainement ressentir lors de la stabilisation de la méthode de génération de colonnes. Par

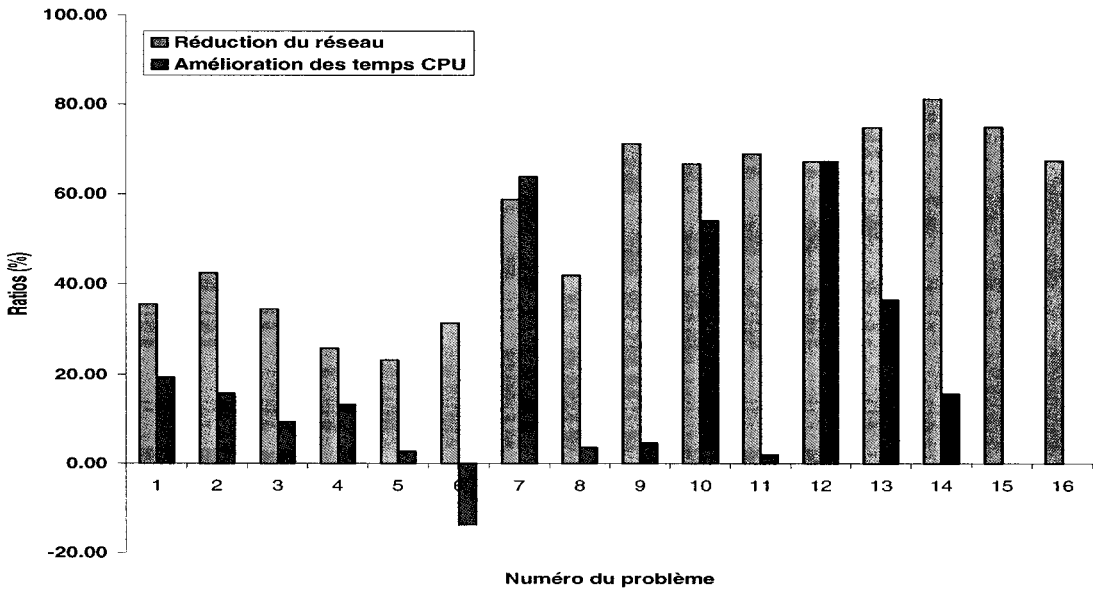

Figure 3.3 – Effet de la réduction du réseau sur les temps CPU totaux

ailleurs, il ne faut pas perdre de vue que la réduction du réseau constitue une étape charnière sur la voie de toute nouvelle investigation, surtout pour la résolution de *MDVSP* en nombres entiers (notre étude étant consacrée uniquement à la relaxation linéaire de ce problème).

CHAPITRE 4 : STABILISATION DE LA PROCÉDURE DE GÉNÉRATION DE COLONNES

Aux chapitres précédents, il a surtout été question de conforter l'hypothèse voulant qu'un problème de tournées de véhicules multi-dépôts à horizon devenait d'autant plus difficile que l'horizon s'allongeait. Bien qu'ayant tenté d'imputer ce comportement à la taille du réseau, il est maintenant établi que cet aspect est loin de constituer la clef de voûte de la problématique. Subséquement, il n'est point de doute que l'augmentation du nombre de tâches par colonne serait l'un des facteurs principaux de la détérioration de la convergence lorsque conjugué à d'autres propriétés de l'algorithme de génération de colonnes.

En effet, les difficultés avec les algorithmes de génération de colonnes apparaissent lorsque ceux-ci sont utilisés pour la résolution de problèmes dégénérés de grande dimension. Il est plutôt admis que la dégénérescence primale, entre autres facteurs, est responsable de l'effet de queue de la génération de colonnes. Néanmoins, l'une des caractéristiques de comportement connues de l'algorithme de génération de colonnes se manifeste de manière aigüe à travers un nombre important d'oscillations des valeurs des variables duales d'une itération à l'autre, donnant lieu à des déplacements de grande amplitude, inutiles et même mauvais, dans l'espace dual réalisable. Ce phénomène, souvent dénommé *instabilité*, se trouve derrière le mauvais conditionnement de l'algorithme. L'instabilité du comportement des valeurs des variables duales est plus fréquente et d'autant plus nocive lorsque les problèmes deviennent plus grands. Au moment où la borne supérieure fournie par la solution primale est non-croissante, la borne inférieure calculée pour chaque point dual n'est pas nécessairement non-décroissante. Aussi, un déplacement d'un bon point dual vers un autre plus mauvais

est toujours possible, et ceci aura pour effet immédiat d'affecter la qualité des colonnes qui seraient générées à l'itération suivante. D'autres aspects de l'instabilité sont discutés en [14].

Pour pallier à ce genre de situations, des approches, dites de *stabilisation*, ont été développées autour d'une idée maîtresse consistant à modifier le problème dual de sorte à guider l'évolution des multiplicateurs dans un sens favorisant la génération de colonnes de meilleure qualité. De telles approches sont à même de restreindre les déplacements inutiles dans l'espace dual qui sont dus au manque d'information sur la fonction duale à l'itération courante.

Dans ce chapitre, nous nous intéressons exclusivement aux techniques de stabilisation de l'algorithme de génération de colonnes basées sur les concepts de région de confiance (*trust region*) et de pénalisation linéaire par morceaux de l'objectif dual. Nous commençons par décrire les grandes lignes de ces méthodes en nous basant sur les récents travaux de recherche effectués par Desrosiers et Ben Amor ([1] et [2]). Nous proposons ensuite des extensions de certains aspects, notamment la structure de la fonction de pénalité et les stratégies de mise à jour des paramètres de stabilisation. Préalablement, pour situer les techniques proposées dans le contexte des travaux de recherche antérieurs, nous consacrons quelques paragraphes à une brève revue de littérature. (Pour une revue de littérature plus étendue, le lecteur peut se référer à [31])

4.1 Revue de littérature

Plusieurs approches de stabilisation ont été proposées pour contrôler et, éventuellement, guider le comportement des valeurs des variables duales le long des itérations de génération de colonnes. Ces approches adoptent généralement l'un des schémas

définis par la classe des *méthodes de type proximal* qui reposent sur l'algorithme du point proximal (*Proximal point algorithm* [29]) et la classe des *méthodes de type faisceaux* qui renvoient vers l'algorithme des faisceaux (*Bundle algorithm* [14, 18]).

Les méthodes de faisceaux (voir [14], [22] et [32]) visent à stabiliser les méthodes de plans coupants en considérant à chaque fois la meilleure solution trouvée, laquelle constitue le centre de stabilité. Le modèle de plans coupants est pénalisé autour du centre de stabilité courant en utilisant la norme Euclidienne et un problème de programmation quadratique à contraintes linéaires est résolu pour déterminer un nouveau point. Au cas où l'amélioration observée de la fonction objectif est jugée suffisante, le centre de stabilité est mis à jour. On parle alors de *pas de descente* (*Serious step*). Au cas contraire, le centre de stabilité reste inchangé et le modèle de plans coupants est enrichi. Dans cette situation, on parle de *pas nul* (*Null step*).

Dans l'algorithme du point proximal (voir [12], [21] et [29]), la fonction objectif du problème est pénalisée loin du *centre de stabilité* en utilisant la norme Euclidienne. Ceci constitue le *problème stabilisé* dont la solution deviendra le prochain centre de stabilité et le nouveau problème stabilisé est encore résolu jusqu'à l'optimalité.

La méthode de Boxstep (voir [24] et [25]) est une méthode proximale dont les fonctions de pénalité contraignent les variables duales à une boîte centrée autour du point dual courant. Les problèmes stabilisés sont résolus par génération de colonnes ou plans coupants. Les auteurs dans [30] utilisent une version des méthodes de faisceaux basée sur le concept de région de confiance pour stabiliser l'algorithme des plans coupants pour des problèmes convexes et non-convexes. Le coefficient de pénalité est mis à jour de manière implicite afin de garantir une valeur adéquate avant d'opérer un pas de descente (serious step ou null step). Kim *et al.* [17] étendent l'idée de l'algorithme du point proximal à l'utilisation de normes linéaires L_1 et L_∞. Les problèmes stabilisés sont résolus par un algorithme de plans coupants stabilisé par

une méthode de type faisceaux pour fournir un meilleur point tout en assurant une amplitude de déplacement minimale et une décroissance significative de la fonction objectif, à moins d'avoir atteint l'optimalité du problème pénalisé. Du Merle *et al.* [10] ont utilisé une fonction de pénalité linéaire à trois morceaux pour pénaliser le modèle des plans coupants. Les auteurs proposent un algorithme de type faisceaux qui tienne compte à la fois des points de vue dual et primal. L'algorithme vise principalement à trouver une solution optimale primale. La fonction de pénalité est conceptuellement similaire aux fonctions de pénalité que nous utilisons ici pour la stabilisation de la procédure de génération de colonnes.

Neame [27] propose un cadre unifié pour les approches de stabilisation dans le cas de la maximisation non contrainte d'une fonction concave linéaire par morceaux. L'analyse de la convergence est basée sur le nombre fini de morceaux de la fonction optimisée. La convergence vers une solution primale optimale n'a pas été investiguée.

Les algorithmes de type proximal garantissent une amélioration stricte de la fonction objectif à l'optimalité des problèmes stabilisés alors que les méthodes de type faisceaux cherchent une amélioration minimale de la fonction objectif à chaque itération de plans coupants. Ces dernières ont besoin d'évaluer la valeur de la fonction objectif du modèle de plans coupants à toute itération. Lorsque le sous-problème est borné, i.e. il n'y a aucun rayon extrême dans le contexte de la décomposition de Dantzig-Wolfe [8], cette évaluation est possible à chaque itération de génération de colonnes ou de plans coupants. Si le sous-problème est non borné, chaque fois qu'une coupe de faisabilité est générée (un rayon extrême dans le contexte de la décomposition de Dantzig-Wolfe [8]), la valeur de l'objectif est $-\infty$ et la qualité du point dual correspondant (et par conséquent de l'amélioration de la valeur de la fonction objectif) ne peut être indiquée avec précision. La situation devient même plus mauvaise lorsque la formulation du problème maître primal ne contient aucune contrainte de convexité explicite. Ce cas s'apparente à la situation où toutes les colonnes à l'exception d'une

seule sont des rayons extrêmes de l'oracle. Ceci est, en effet, le cas de l'approche classique de génération de colonnes. En fait, tous les points duaux sauf celui obtenu à la toute dernière itération sont non réalisables, donc les valeurs des bornes inférieures qui leur correspondent sont $-\infty$. Lorsqu'ils sont appliqués dans cette situation, les algorithmes de faisceaux sont capables d'opérer des pas de descente seulement à défaut de génération de colonnes de coût réduit négatif, en d'autres termes, à l'optimalité des problèmes stabilisés. Dans ce cas, ils se comportent de manière similaire aux algorithmes proximaux.

Les algorithmes abordés dans la présente étude sont de type proximal, combinant les concepts *pénalité* et *région de confiance* ([1] et [2]). Les points duaux peuvent prendre, sans le moindre coût, des valeurs dans un ensemble convexe fermé de pleine dimension autour de l'estimation duale courante (centre de stabilité) au moment où des pénalités sont imputées en dehors de cette région de confiance.

4.2 Idées de base de la stabilisation par une fonction de pénalité linéaire par morceaux

L'approche de stabilisation par une fonction de pénalité linéaire par morceaux s'applique à tout programme linéaire. Néanmoins, celle-ci revêt une importance particulière lorsque les problèmes à résoudre sont difficiles. Les problèmes résolus par la procédure de génération de colonnes constituent l'exemple typique de ces problèmes, lesquels font intervenir un nombre impressionnant de colonnes sans que ces colonnes ne puissent être toutes connues d'avance.
Étant donné une matrice A de dimension $m \times n$, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ et $x \in \mathbb{R}^n$, notons par (P) le programme primal et (D) son dual.
Les formulations générales, sous forme matricielle, de (P) et (D) sont :

(P)	(D)
$Min\ c^T x$	$Max\ b^T \pi$
$Ax = b$	$A^T \pi \leq c$
$x \geq 0$	$\pi \in \mathbb{R}^m$

La procédure de génération de colonnes appliquée à (P) et (D) consiste à considérer seulement un sous-ensemble de colonnes à la fois. À chaque itération ℓ de génération de colonnes, un problème maître restreint (P_ℓ) et son dual (D_ℓ) sont résolus, lesquels sont définis par :

(P_ℓ)	(D_ℓ)
$Min\ c_\ell^T x_\ell$	$Max\ b^T \pi$
$A_\ell x_\ell = b$	$A_\ell^T \pi \leq c_\ell$
$x_\ell \geq 0$	$\pi \in \mathbb{R}^m$

où c_ℓ et A_ℓ sont respectivement le vecteur coût et la matrice définis avec l'ensemble des variables courantes x_ℓ.
L'approche de stabilisation est appliquée au problème dual (D) mais, dès lors que nous nous intéressons aussi à une solution optimale primale pour (P), les deux problèmes sont considérés simultanément.

Les idées principales autour desquelles pivote l'approche de stabilisation sont décrites ci-après.

- Construire une pénalité linéaire autour d'une estimation de la solution optimale duale (centre de stabilité). Cette option suppose que le centre de stabilité renferme suffisamment d'information pour se placer dans un voisinage de l'optimum dual à même de limiter les déplacements inutiles dans l'espace dual.

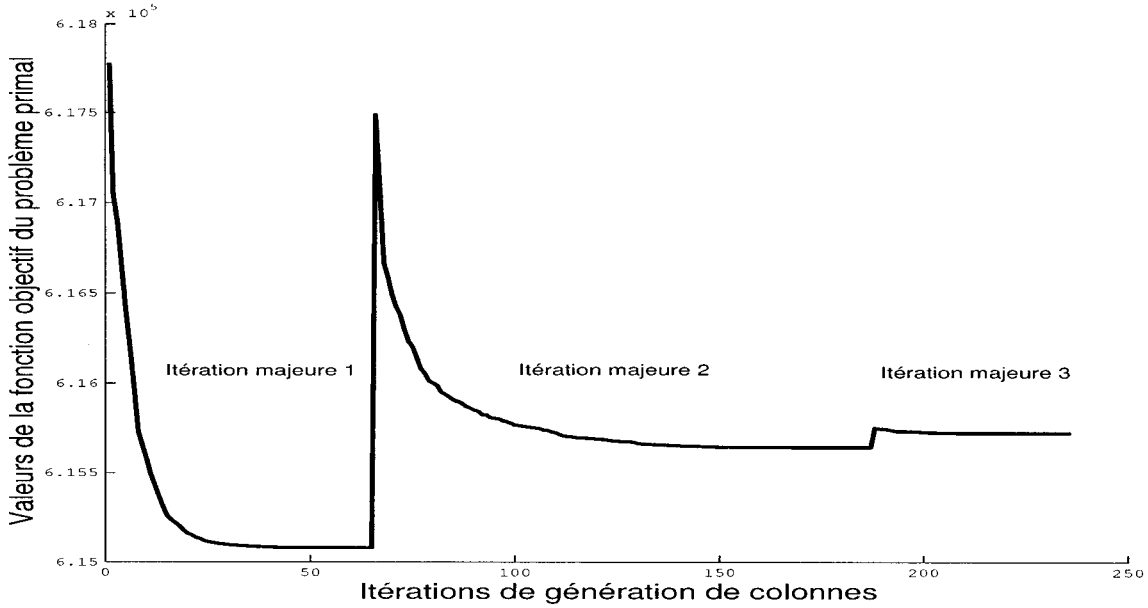

Figure 4.1 – Processus de stabilisation à trois itérations majeures

- Appliquer cette pénalité linéaire au problème dual (D) pour former une paire de problèmes stabilisés (pénalisés) $((SP_l), (SD_l))$. L'indice l représente l'*itération majeure* courante, laquelle correspond à l'application d'une procédure de génération de colonnes complète pour résoudre les problèmes stabilisés.
- Après chaque itération majeure l, effectuer une remise à jour des coefficients de la fonction de pénalité, souvent identifiés comme *paramètres de stabilisation*, moyennant *une stratégie de mise à jour* qui garantisse la convergence de la suite de problèmes stabilisés $((SP_l), (SD_l))$ ainsi générée vers des solutions optimales de (P) et de (D).

La figure 4.1 montre l'évolution de l'objectif primal dans un processus à trois itérations majeures. La valeur optimale de chaque itération est une borne inférieure sur la valeur optimale de (P). La figure illustre la croissance stricte de cette borne avec les itérations majeures.

Avec l l'indice de l'itération majeure courante, nous présentons la fonction de péna-

lité par morceaux $-g^l(\pi)$ ($\pi = [\pi_1, \ldots, \pi_m]^T$) sous la forme définie par Ben Amor [1] ainsi que la paire de problèmes stabilisés y afférant $((SP_l), (SD_l))$, avant de nous pencher sur le cas spécifique qui nous intéresse.

4.3 Fonction de pénalité linéaire à cinq morceaux

L'approche discutée dans [1] et [2] est basée sur le principe de région de confiance autour d'un centre de stabilité à l'extérieur de laquelle une pénalité est appliquée. La fonction de pénalité comporte cinq morceaux et présente la particularité que la région de confiance est un pavé non-vide de pleine dimension contenant le centre de stabilité dans son intérieur. Il a été démontré que si une solution optimale duale est connue, la résolution des problèmes stabilisés autour de cette solution assure l'obtention d'une solution optimale primale de base. La convergence de l'algorithme vers une paire de solutions optimales primale et duale a également été établie.

4.3.1 Définition

Soit π le vecteur des variables duales associées au problème dual (D) défini à la section 4.2. La fonction de pénalité linéaire à cinq morceaux $-g^l(\pi)$ s'exprime en termes de ses composantes $g_i^l(\pi_i)$ $(i = 1, \ldots, m)$ par :

$$g^l(\pi) = \sum_{i=1}^{m} {g_i}^l(\pi_i) \tag{4.1}$$

où

$$g_i^l(\pi_i) = \begin{cases} (\zeta_{-,i}^l + \varepsilon_{-,i}^l)(\gamma_{-,i}^l - \pi_i) & si \quad -\infty \leq \pi_i \leq \gamma_{-,i}^l \\ \varepsilon_{-,i}^l(\delta_{-,i}^l - \pi_i) & si \quad \gamma_{-,i}^l \leq \pi_i \leq \delta_{-,i}^l \\ 0 & si \quad \delta_{-,i}^l \leq \pi_i \leq \delta_{+,i}^l \\ \varepsilon_{+,i}^l(\pi_i - \delta_{+,i}^l) & si \quad \delta_{+,i}^l \leq \pi_i \leq \gamma_{+,i}^l \\ (\varepsilon_{+,i}^l + \zeta_{+,i}^l)(\pi_i - \gamma_{+,i}^l) & si \quad \gamma_{+,i}^l \leq \pi_i \leq +\infty \end{cases}$$

où $\gamma^l_.$, $\delta^l_.$, $\varepsilon^l_.$ et $\zeta^l_.$ sont les paramètres de stabilisation

Afin que la définition de la fonction soit cohérente, on doit avoir :

$$\gamma^l_- \leq \delta^l_- \leq \delta^l_+ \leq \gamma^l_+. \tag{4.2}$$

Pour que $g^l_i (i = 1, \ldots, m)$ soit concave, les conditions suivantes doivent être vérifiées :

$$\zeta^l_- > \varepsilon^l_- > 0$$
$$\zeta^l_+ > \varepsilon^l_+ > 0.$$

La région de confiance correspond au pavé défini par les vecteurs δ^l_- et δ^l_+ i.e. l'ensemble $\{\pi \in \mathbb{R}^m : \delta^l_- \leq \pi \leq \delta^l_+\}$. Ainsi, pour chaque composante π_i, on tolère sans pénalité les valeurs à l'intérieur de l'intervalle $[\delta^l_-, \delta^l_+]$. Une pénalié égale à $-\varepsilon^l_{+,i}$ (respectivement $-\varepsilon^l_{-,i}$) est appliquée dans l'intervalle $[\delta^l_{+,i}, \gamma^l_{+,i}]$ (respectivement $[\gamma^l_{-,i}, \delta^l_{-,i}]$). Pour $\pi_i > \gamma^l_{+,i}$ (respectivement $\pi_i < \gamma^l_{-,i}$), la pénalité est $-(\varepsilon^l_{+,i} + \zeta^l_{+,i})$ (respectivement $-(\varepsilon^l_{-,i} + \zeta^l_{-,i})$). La figure 4.2 illustre la fonction $g^l_i(\pi)$ autour de la composante $\hat{\pi}^l_i$ du centre de stabilisation $\hat{\pi}^l$.

4.3.2 Formulation des problèmes stabilisés

Le problème dual pénalisé (SD_l) est formulé comme suit :

$$Max\ b^T\pi - g^l(\pi)$$
$$A^T\pi \leq c.$$

La fonction $g^l(\pi)$ étant linéaire par morceaux et convexe, (SD_l) peut être écrit comme un programme linéaire. Son dual est appelé problème primal stabilisé (SP_l). Les

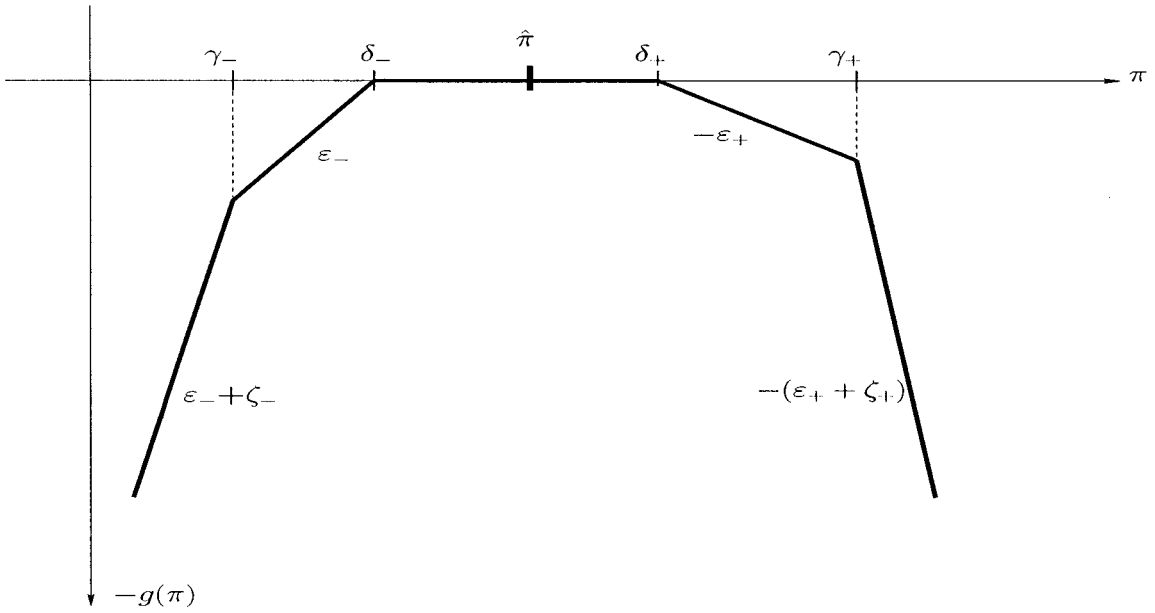

Figure 4.2 – Fonction de pénalité linéaire à cinq morceaux

formulations des deux problèmes sont les suivantes :

$$(SD_l)$$

$$\begin{aligned} Max\ & b^T\pi - \zeta_-^T v^- - \varepsilon_-^T u^- - \varepsilon_+^T u^+ - \zeta_+^T v^+ \\ & A^T\pi \leq c \\ & \delta_- - u^- \leq \pi \leq \delta_+ + u^+ \\ & \gamma_- - v^- \leq \pi \leq \gamma_+ + v^+ \\ & u^-, v^-, u^+, v^+ \geq 0. \end{aligned}$$

$$(SP_l)$$

$$\begin{aligned} Min\ & c^T x - \gamma_-^T z^- - \delta_-^T y^- + \delta_+^T y^+ + \gamma_+^T z^+ \\ & Ax - z^- - y^- + y^+ + z^+ = b \\ & z^- \leq \zeta_-, \qquad y^- \leq \varepsilon_- \\ & y^+ \leq \varepsilon_+, \qquad z^+ \leq \zeta_+ \\ & x, z^-, y^-, y^+, z^+ \geq 0. \end{aligned}$$

4.4 Fonction de pénalité linéaire à quatre morceaux

Il y a lieu de noter que la fonction de pénalité linéaire à quatre morceaux n'est autre qu'un cas particulier de la fonction de pénalité linéaire à cinq morceaux pour laquelle les valeurs respectives de δ_- et γ_- s'annulent. Autrement, au lieu de se situer à l'intérieur de la région de confiance, le centre de stabilisation "colle" désormais au bord de cet intervalle, tel qu'illustré par la figure 4.3. Par conséquent, la formulation de la fonction de pénalité sera légérement modifiée, de même que celle des problèmes stabilisés.

4.4.1 Définition

Comme précédemment, la fonction de pénalité linéaire à quatre morceaux $-g^l(\pi)$ se définit en termes de ses composantes $g_i^l(\pi_i)$ $(i = 1, \ldots, m)$ par :

$$g^l(\pi) = \sum_{i=1}^{m} {g_i}^l(\pi_i) \tag{4.3}$$

où

$$g_i^l(\pi_i) = \begin{cases} -\pi_i \ \varepsilon_{-,i}^l & si \quad -\infty \leq \pi_i \leq \hat{\pi}_i^l \\ 0 & si \quad \hat{\pi}_i^l \leq \pi_i \leq \delta_{+,i}^l \\ \varepsilon_{+,i}^l(\pi_i - \delta_{+,i}^l) & si \quad \delta_{+,i}^l \leq \pi_i \leq \gamma_{+,i}^l \\ (\varepsilon_{+,i}^l + \zeta_{+,i}^l)(\pi_i - \gamma_{+,i}^l) & si \quad \gamma_{+,i}^l \leq \pi_i \leq +\infty. \end{cases}$$

Toujours, pour des besoins de cohérence de la définition, on doit avoir :

$$\gamma_+^l \geq \delta_+^l \geq \hat{\pi}^l. \tag{4.4}$$

Par contre, pour la concavité de $g_i^l (i = 1, \ldots, m)$, les conditions suivantes doivent être vérifiées :

$$\varepsilon_-^l > 0$$
$$\zeta_+^l > \varepsilon_+^l > 0.$$

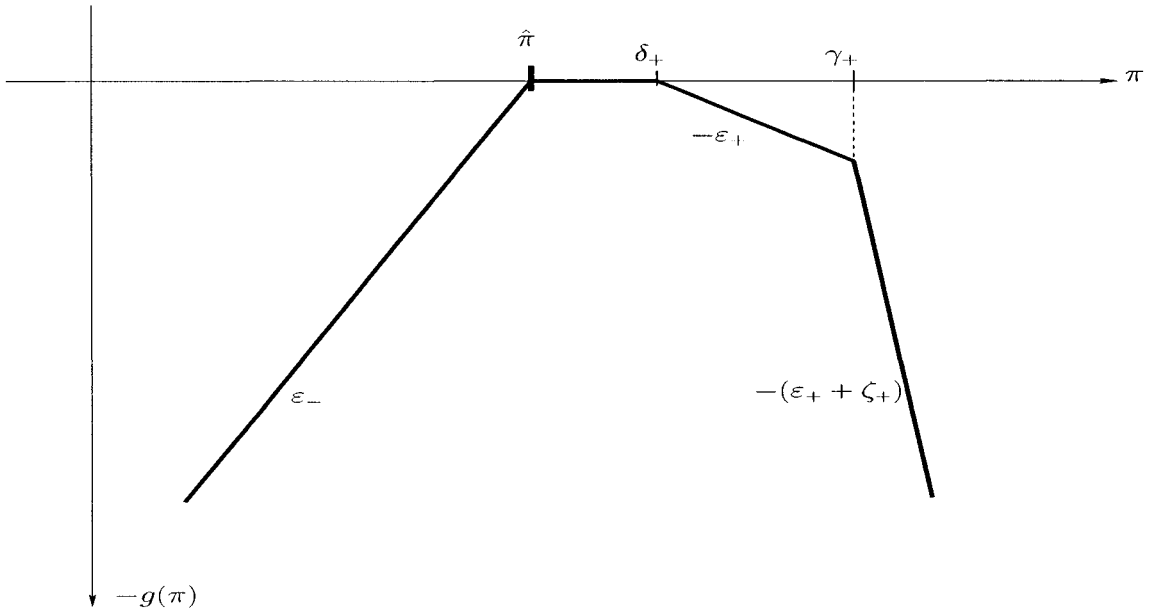

Figure 4.3 – Fonction de pénalité linéaire à quatre morceaux

Ici, la région de confiance correspond au pavé défini par l'ensemble $\{\pi \in \mathbb{R}^m : \hat{\pi}_i^l \leq \pi \leq \delta_+^l\}$. Là aussi, pour chaque composante π_i, on tolère sans pénalité les valeurs à l'intérieur de l'intervalle $[\hat{\pi}_i^l, \delta_+^l]$. Une pénalité égale à $-\varepsilon_{+,i}^l$ est appliquée dans l'intervalle $[\delta_{+,i}^l, \gamma_{+,i}^l]$. Pour $\pi_i > \gamma_{+,i}^l$, la pénalité est $-(\varepsilon_{+,i}^l + \zeta_{+,i}^l)$ alors que, pour $\pi_i \leq \hat{\pi}_i^l$, la pénalité est $-\varepsilon_{-,i}^l$

4.4.2 Formulation des problèmes stabilisés

Les formulations des deux problèmes primal et dual stabilisés, (SP_l) et (SD_l) respectivement, sont les suivantes :

$$(SD_l)$$

$$\begin{aligned} Max\ & b^T\pi - \varepsilon_-^T u^- - \varepsilon_+^T u^+ - \zeta_+^T v^+ \\ & A^T\pi \leq c \\ & -u^- \leq \pi \leq \delta_+ + u^+ \\ & \pi \leq \gamma_+ + v^+ \\ & u^-, u^+, v^+ \geq 0 \end{aligned}$$

$$(SP_l)$$

$$\begin{aligned} Min\ & c^T x + \delta_+^T y^+ + \gamma_+^T z^+ \\ & Ax - z^- - y^- + y^+ + z^+ = b \\ & y^- \leq \varepsilon_-, \qquad y^+ \leq \varepsilon_+, \qquad z^+ \leq \zeta_+ \\ & x, y^-, y^+, z^+ \geq 0. \end{aligned}$$

4.5 Procédure de stabilisation

Ci-dessous sont décrites, de manière explicite, les principales étapes de l'approche de stabilisation proposée. Nous ferons de sorte à nous attarder sur des points de détail que nous estimons d'un meilleur intérêt pratique et, peut-être potentiellement objets d'approfondissements futurs tandis que nous passerons sur d'autres points en n'en citant que l'intitulé, ceux-ci supposés triviaux au stade de notre étude. Il est utile de rappeler que, ce qui caractérise l'algorithme de stabilisation, est qu'il se déroule sous une forme qui permet de distinguer deux types d'itérations : majeures et mineures. Une itération majeure l est le processus de résolution de la paire de problèmes $((SP_l), (SD_l))$ à l'optimalité par la méthode de génération de colonnes, chaque itération de génération de colonnes représentant toute seule une itération

mineure. À la fin d'une itération majeure l, si x^l est réalisable pour (P), l'optimalité est atteinte et l'algorithme s'arrête. L'algorithme s'arrête aussi si l'un des problèmes (SP_l) ou (SD_l) est non-réalisable ou non-borné. Si les deux problèmes sont réalisables et finis, mais que l'optimalité n'est pas atteinte, le centre de stabilisation ainsi que la fonction de pénalisation sont mis à jour.
D'abord, voici les étapes essentielles de l'algorithme :

Étape 0 : *Initialisation*
Définir $\hat{\pi}^0$ et $g^l(\pi)$; $l = 0$.

Étape 1 : *Itération majeure l*
Résoudre $((SP_l), (SD_l))$ par génération de colonnes.

1. Si (SP_l) et (SD_l) sont réalisables et finis, alors
   - Si les variables de pénalisation $z_{-,l}$, $y_{-,l}$, $y_{+,l}$, et $z_{+,l}$ sont toutes nulles, alors l'optimum est atteint pour (P) et (D). STOP
   - Sinon, passer à l'étape suivante
2. Si (SP_l) et (SD_l) sont non réalisables et non bornés. STOP

Étape 2 : *Déplacement du centre de stabilisation et mise à jour des paramètres*

1. $\hat{\pi}^{l+1} = \pi^l$.
2. Définir une stratégie de mise à jour des valeurs des paramètres de stabilisation $\delta_+^{l+1}, \delta_-^{l+1}, \gamma_+^{l+1}, \gamma_-^{l+1}, \varepsilon_+^{l+1}, \varepsilon_-^{l+1}, \zeta_+^{l+1}$ et ζ_-^{l+1}.
3. Définir $g^{l+1}(\pi)$.
4. Reprendre le processus à partir de l'étape 1.

4.5.1 Initialisation

Cette étape de l'algorithme est décisive pour les considérations théoriques longuement abordées en début de chapitre et ailleurs, mais aussi pour des raisons pratiques

auxquelles des explications seraient certainement nécessaires.
Les entités concernées par l'initialisation sont :

Le vecteur des valeurs initiales des variables duales $\hat{\pi}^0$:
L'algorithme converge quelque soit le vecteur initial $\hat{\pi}^0$. Même si $\hat{\pi}^0$ n'est pas réalisable, $\hat{\pi}^l$ sera réalisable pour tout $l \geq 1$. Le choix d'un vecteur initial assez proche de l'optimum permet d'accélérer la résolution de manière significative. Par contre, un mauvais choix nécessiterait plus d'itérations majeures et serait plus coûteux en temps CPU [1]. Pour ce qui est de la présente application, nous préconisons l'estimation de $\hat{\pi}^0$.

Tel que mentionné au chapitre précédent, l'écart entre la valeur optimale de la relaxation lagrangienne du problème *MDVSP* et la solution optimale entière du problème *SDVSP* a tendance à se rétrécir à mesure que l'horizon s'allonge. Ce qui nous permet de penser que la composition des tournées optimales du problème *MDVSP* se rapprocherait de plus en plus de celle du problème *SDVSP* pour des horizons relativement longs. Pareil comportement place la solution duale du problème *SDVSP* en meilleure position pour l'initialisation des variables duales qui entrent dans l'expression de la fonction de pénalité initiale, l'obtention de cette solution n'étant pas coûteuse en temps CPU.

Comme il ne faut pas aussi perdre de vue que la solution duale du problème *SDVSP* est, jusqu'à ce stade, un résultat auxilliaire de la phase de réduction du réseau (*cf.* Chapitre 3), nous rappellerons que nous avons opéré une *double réduction*, ce qui signifie que nous disposons désormais de deux solutions duales différentes pour un même problème *SDVSP*, à savoir : $\hat{\pi}_s$ et $\hat{\pi}_{sr}$. Par conséquent, un plus large éventail de solutions duales $\hat{\pi}^0$ est possible en prenant des combinaisons convexes de $\hat{\pi}_s$ et $\hat{\pi}_{sr}$, soit :

$$\hat{\pi}^0 = \alpha\hat{\pi}_s + (1-\alpha)\hat{\pi}_{sr} \qquad \alpha \in [0,1].$$

Bien que notre application ait été limitée aux seuls cas où $\alpha \in \{0, 0.5, 1\}$, les résultats obtenus méritent un examen particulier (nous y reviendrons au cours de l'exposé). Nous noterons $\hat{\pi}_m$ le vecteur des valeurs duales moyennes calculées pour $\alpha = 0.5$.

Les paramètres de stabilisation initiaux :

Au regard de l'expression de la fonction de pénalité, le choix des valeurs initiales de δ, γ, ζ et ε ne manque pas d'importance pour juger de l'efficacité de l'algorithme. Ce choix est déterminant dans la mesure où ces paramètres influencent non seulement la complexité du processus de résolution des problèmes stabilisés, mais aussi le nombre d'itérations majeures requises pour l'obtention des optima des problèmes originaux. Ces paramètres vont être abordés dans l'ordre grandissant de la difficulté rencontrée, en pratique, pour en estimer la valeur.

1. ***Estimation de la largeur de la région de confiance***

La région de confiance correspond à un ensemble défini autour du centre de stabilité courant où aucune pénalité n'est appliquée. À l'extérieur de cette région, une pénalité est encourue. Le but est d'empêcher le prochain centre de stabilité de trop s'éloigner du centre courant tout en procurant une certaine flexibilité à l'algorithme. En fait, il s'agit tout simplement d'estimer les valeurs de δ_+ et δ_- ou encore la seule valeur de δ_+, suivant le nombre de morceaux considérés dans la fonction de pénalité linéaire. Comme l'intervalle est généralement choisi symétrique par rapport à $\hat{\pi}^0$, il suffira donc d'estimer uniquement δ_+, qu'importe le nombre de morceaux.

Ben Amor [1] affirme que l'utilisation d'intervalles trop petits résulterait en un très grand nombre d'itérations majeures, lesquelles permettraient des déplacements minimes du centre de stabilité. Ceci conduirait ainsi à une convergence lente du processus de résolution, allant jusqu'à devenir moins efficace que la procédure de génération de colonnes standard. D'un autre côté, si l'intervalle de confiance est trop grand, le problème est généralement résolu au complet au bout d'une seule itération majeure, ramenant la stabilisation à la méthode de génération de colonnes standard.

À travers notre application, il nous a été offert de vérifier que ce facteur seul ne justifie pas toujours ce comportement sans y adjoindre les spécificités du problème à résoudre dont dépendent les valeurs allouées aux bornes supérieures et inférieures sur les paramètres de stabilisation. Cet aspect sera mis plus en relief lorsque nous développerons le plan d'expérimentation. Pratiquement, l'unique moyen à portée de la main pour accomplir cette tâche reste la recherche par tâtonnement, combinée à des astuces de calcul simples qui font intervenir le nombre de tâches du problème ainsi que les écarts existant entre les optima disponibles. Au demeurant, il est à noter que plusieurs largeurs d'intervalles ont été testées sur les 16 problèmes *MDVSP* à horizon objets de la présente étude et, malgré la disparité qui caractérise ces derniers (mis à part le nombre de tâches et de dépôts), tous les processus de stabilisation ont été conduits de manière efficace avec des valeurs de δ relativement petites, se situant entre 0.0001 et 1.

2. ***Estimation de la largeur de la région de pénalisation***

Cette région se situe à l'extérieur de l'intervalle de confiance et constitue la zone où seront appliquées les pénalités ε qui empêchent des déplacements inutiles loin du centre de stabilisation. Il s'agira, notamment, d'estimer les valeurs de γ_- et γ_+ pour la cas d'une fonction de pénalité linéaire à cinq morceaux et γ_+ seule pour une pénalité à quatre morceaux. Pour les problèmes particuliers que nous avons traités, cette valeur a été choisie dans une fourchette de 0.001 à 1.

3. ***Estimation des pénalités***

Des pénalités trop élevées limitent le déplacement du centre de stabilité à l'extérieur de la région de confiance alors que des pénalités faibles facilitent ce déplacement. Par exemple, si ε_- et ε_+ sont faibles, la fonction de pénalité permet d'élargir l'intervalle de confiance mais avec une légère perturbation. Par ailleurs, pour que la pénalité ait un effet de restriction, il faut que la fonction de pénalité utilisée g^l vérifie pour au moins un certain $i \in \{1, \ldots, m\}$ [2]

$$\varepsilon_{-,i} + \zeta_{-,i} > 1$$

$$\varepsilon_{+,i} + \zeta_{+,i} > 1.$$

Tenant compte de ces conditions supplémentaires, nous avons fixé à 0.1 la valeur de ε et à 1.0 la valeur de ζ pour tous les tests effectués.
De la sorte, nous aurons défini tous les paramètres de $g^l(\pi)$ à l'itération $l = 0$.

4.5.2 Stratégies de mise à jour des paramètres de stabilisation

Pour la mise à jour des paramètres de stabilisation, plusieurs stratégies sont possibles, souvent basées sur des idées intuitives [2]. Dans l'algorithme proposé, le nouveau centre de stabilisation $\hat{\pi}^{l+1}$ est toujours la solution optimale π^l de (SD_l). Cependant, on tient compte seulement des paramètres de la fonction g^l dans l'itération majeure suivante. En effet, ce qui est important est que π^l soit à l'intérieur de la boîte $]\delta_-^l, \delta_+^l[$. Cette dernière condition peut n'être imposée qu'à partir d'un certain nombre fini d'itérations majeures. Dans ce qui suivra, nous décrivons les stratégies testées.

Stratégies *dynamiques* :
La stratégie *dynamique* tend à modifier les paramètres de la fonction de pénalité en fonction de la position de la solution du problème stabilisé par rapport aux intervalles délimitant les régions de confiance ainsi que les régions de pénalisation. Pour la mise en application de l'algorithme de stabilisation, nous avons testé différentes stratégies *dynamiques*, dont la description vient ci-dessous. Les intervalles de confiance et les pénalités sont réduits ou élargis par un facteur 2, selon les circonstances. Toutefois, ceci n'exclut en rien la possibilité de tester d'autres facteurs d'élargissement ou de réduction autres que 2.

1. ***Stratégie dynamique bilatérale*** : opérationnelle avec une fonction de pénalité linéaire à cinq morceaux, cette stratégie adopte la logique suivante :
   - Si la valeur de la solution duale courante π_i^l se trouve à l'intérieur de l'intervalle de confiance, un recentrage autour du nouveau centre de stabilité π_i^l prend place, accompagné d'une réduction de la boîte $[\delta_{-,i}^l, \delta_{+,i}^l]$ avec une accentuation de la fermeture par une augmentation des pénalités ε, ce que l'on peut traduire par

$$\text{Si } \pi_i^l \in]\delta_{-,i}^l, \delta_{+,i}^l[\text{ alors } \begin{cases} \delta_{+,i}^{l+1} = \delta_{+,i}^l/2 & \text{et} \quad \delta_{-,i}^{l+1} = \delta_{-,i}^l/2 \\ \varepsilon_{+,i}^{l+1} = \varepsilon_{+,i}^l \times 2 & \text{et} \quad \varepsilon_{-,i}^{l+1} = \varepsilon_{-,i}^l \times 2. \end{cases}$$

   - Si, par contre, π_i^l se retrouvait en dehors de l'intervalle de confiance, ce dernier se verrait élargi tout en diminuant la pénalité à subir, de sorte que :

$$\text{Si } \pi_i^l \geq \delta_{+,i}^l, \text{ alors } \begin{cases} \delta_{+,i}^{l+1} = \delta_{+,i}^l \times 2 \\ \varepsilon_{+,i}^{l+1} = \varepsilon_{+,i}^l/2. \end{cases}$$

$$\text{Si } \pi_i^l \leq \delta_{-,i}^l, \text{ alors } \begin{cases} \delta_{-,i}^{l+1} = \delta_{-,i}^l \times 2 \\ \varepsilon_{-,i}^{l+1} = \varepsilon_{-,i}^l/2. \end{cases}$$

2. ***Stratégie dynamique unilatérale simple*** : appliquée principalement lorsque la fonction de pénalité comporte quatre morceaux, le seul aspect distinctif de cette stratégie réside dans le fait que l'intervalle de confiance est unilatéral, par opposition au cas précédent. Par conséquent, nous avons envisagé la mise à jour des paramètres de stabilisation conformément à ce qui suit :
   - Si la valeur de π_i^l tombe à l'intérieur de l'intervalle de confiance, le centre de stabilité est déplacé vers la position de π_i^l alors que l'amplitude de l'intervalle se voit réduite de moitié et une plus grande pénalité est appliquée de part et d'autre de l'intervalle.

$$\text{Si } \pi_i^l \in]\hat{\pi}_i^l, \delta_{+,i}^l[\text{ alors : } \begin{cases} \delta_{+,i}^{l+1} = \delta_{+,i}^l/2 \\ \varepsilon_{+,i}^{l+1} = \varepsilon_{+,i}^l \times 2 \quad \text{et} \quad \varepsilon_{-,i}^{l+1} = \varepsilon_{-,i}^l \times 2. \end{cases}$$

- Dans le cas contraire, les modifications à introduire varient en fonction de l'emplacement de la valeur de $\hat{\pi}_i^l$.

$$\text{Si } \pi_i^l \geq \delta_{+,i}^l, \text{ alors } \begin{cases} \delta_{+,i}^{l+1} = \delta_{+,i}^l \times 2 \\ \varepsilon_{+,i}^{l+1} = \varepsilon_{+,i}^l / 2. \end{cases}$$

$$\text{Si } \pi_i^l \leq \hat{\pi}_i^{l-1}, \text{ alors } \{ \ \varepsilon_{-,i}^{l+1} = \varepsilon_{-,i}^l / 2.$$

3. ***Stratégie dynamique unilatérale forcée*** : procédant exactement de la même manière que la stratégie dynamique unilatérale simple, le schéma de réduction que celle-ci adopte, à partir de la deuxième itération majeure, ne fait plus intervenir l'amplitude de l'intervalle de confiance seule mais aussi l'écart entre le nouveau centre de stabilité et le précédent. Nous avons analysé ce scénario sous cet angle :

$$\text{Si } \pi_i^l \in]\hat{\pi}_i^l, \delta_{+,i}^l[, \text{ alors } \begin{cases} \delta_{+,i}^{l+1} = (\delta_{+,i}^l + (\pi_i^l - \hat{\pi}_i^l))/2 \\ \varepsilon_{+,i}^{l+1} = \varepsilon_{+,i}^l \times 2 \\ \varepsilon_{-,i}^{l+1} = \varepsilon_{-,i}^l \times 2. \end{cases}$$

$$\text{Si } \pi_i^l \geq \delta_{+,i}^l, \text{ alors } \begin{cases} \delta_{+,i}^{l+1} = \delta_{+,i}^l \times 2 \\ \varepsilon_{+,i}^{l+1} = \varepsilon_{+,i}^l / 2. \end{cases}$$

$$\text{Si } \pi_i^l \leq \hat{\pi}_i^l, \text{ alors } \begin{cases} \delta_{+,i}^{l+1} = (\hat{\pi}_i^l - \pi_i^l)/2 \\ \varepsilon_{-,i}^{l+1} = \varepsilon_{-,i}^l / 2. \end{cases}$$

En fait, l'idée principale consiste à se positionner par rapport au précédent centre de stabilité et faire en sorte à forcer le déplacement de la borne supérieure du nouvel intervalle de confiance du côté dicté par l'emplacement du nouveau centre de stabilité.

Stratégie *hybride* :

Cette stratégie opère des modifications limitées aux paramètres de stabilisation et

garde la fonction de pénalité inchangée à partir d'une certaine étape du processus. La stratégie hybride consiste à applatir la fonction dans la région de pénalité ε après chaque itération majeure en bornant ε inférieurement à 10^{-4}. Les intervalles $[\delta_{\pm}, \gamma_{\pm}]$ sont élargis si π_l, solution duale optimale obtenue à la fin de l'itération majeure courante, se retrouve à l'extérieur de l'intervalle $[\gamma_-, \gamma_+]$. D'un point de vue expérimental, les résultats des tests s'avèrent concluants pour un facteur d'élargissement de 10.

4.6 Conclusion

Le chapitre en cours a exclusivement été consacré aux aspects théoriques autour desquels gravite la notion de stabilisation de la procédure de génération de colonnes. En effet, nous avons passé en revue l'essentiel des techniques qui ont été développées dans le domaine, avant de nous pencher sur les méthodes proximales, famille à laquelle appartient l'approche que nous avons discutée. Une fois les grandes lignes de notre méthode exposées, nous avons détaillé les principaux points de l'algorithme de stabilisation, dont l'initialisation et les stratégies de mise à jour des paramètres de stabilisation. Au stade où notre étude a été menée, nous pensons avoir cerné l'ensemble des concepts théoriques pour être en mesure d'entamer une mise en œuvre des techniques discutées en matière de stabilisation de la procédure de génération des colonnes.

CHAPITRE 5 : APPLICATION DE LA STABILISATION AU *MDVSP* À HORIZON LONG

La stabilisation de la procédure de génération des colonnes ne saurait être consistante sans une étude expérimentale préliminaire visant à la fois à amoindrir les difficultés susceptibles d'être générées par l'extension de l'horizon et, surtout, à tester une méthodologie d'approche pour la généralisation des caractéristiques initiales des paramètres de stabilisation.

À cet effet, nous commençons par présenter les formulations mathématiques des problèmes stabilisés associés au *MDVSP* à horizon long. Par la suite, nous développons une approche préliminaire utile pour forger une idée intuitive quant à l'ordre de grandeur des paramètres de stabilisation dans le contexte particulier des problèmes que nous résolvons. Pour l'application proprement dite, en plus de la stabilisation des variables duales, d'autres techniques d'accélération de la procédure de génération de colonnes sont aussi testées conjointement avec celles abordées au chapitre quatrième.

Par ailleurs, l'analyse des résultats est entreprise à deux niveaux : À un premier niveau, en moyenne, pour mesurer l'effet des facteurs, autres que les paramètres de stabilisation, sur les performances de la procédure de stabilisation. À un autre niveau, l'analyse est effectuée sur la base des meilleurs résultats produits pour situer, s'il y a lieu, toute corrélation entre les performances de la procédure de stabilisation et le choix de paramètres de stabilisation initiaux.

5.1 Formulation des problèmes stabilisés

L'approche de stabilisation portera uniquement sur les variables duales associées aux contraintes de couverture dès lors que celles-ci se présentent en un nombre relativement élevé, par opposition aux contraintes sur le nombre de véhicules, constituant, de fait, la source principale de la difficulté du problème. D'un autre côté, la génération même des problèmes test suppose un nombre de véhicules suffisant pour couvrir l'ensemble des tâches, de sorte que les multiplicateurs des contraintes correspondantes sont nuls à l'optimalité.
Nous présentons ici les formulations du problème maître primal stabilisé (SMP_l) ainsi que son dual (SMD_l) à une itération majeure l donnée pour les cas d'une fonction de pénalité linéaire à cinq morceaux [2] ensuite d'une fonction linéaire à quatre morceaux.

5.1.1 Cas d'une fonction de pénalité linéaire à cinq morceaux

Lorsque la fonction de pénalité est linéaire à cinq morceaux, les problèmes stabilisés se formulent comme suit :

$$(SMD_l)$$

$$Max \sum_{i=1}^{m}(b_i\pi_i - \zeta_{-,i}v_i^- - \varepsilon_{-,i}u_i^- - \varepsilon_{+,i}u_i^+ - \zeta_{+,i}v_i^+) - \sum_{k\in K} n_k\lambda^k$$

$$\begin{array}{rcccll}
\sum_{i=1}^{m} a_{ip}\pi_i & - & \sum_{k\in K} b_p^k\lambda^k & \leq & c_p, & \forall p \in \Omega \\
\delta_{-,i} - u_i^- & \leq & \pi_i & \leq & \delta_{+,i} + u_i^+, & i = 1,\ldots,m \\
\gamma_{-,i} - v_i^- & \leq & \pi_i & \leq & \gamma_{+,i} + v_i^+, & i = 1,\ldots,m \\
u^-, v^-, & \lambda, & u^+, v^+ & \geq & 0. &
\end{array}$$

$$(SMP_l)$$

$$Min \sum_{p\in\Omega} c_p\theta_p + \sum_{i=1}^{m}(-\gamma_{-,i}^l z_i^- - \delta_{-,i}^l y_i^- + \delta_{+,i}^l y_i^+ + \gamma_{+,i}^l z_i^+)$$

$$\begin{array}{rcll}
\sum_{p\in\Omega} a_{ip}\theta_p - z_i^- - y_i^- + y_i^+ + z_i^+ & = & 1, & i = 1,\ldots,m \\
\sum_{p\in\Omega} b_p^k\theta_p & \leq & n_k, & k = 1,\ldots,|K| \\
z^- & \leq & \zeta_- & \\
y^- & \leq & \varepsilon_- & \\
y^+ & \leq & \varepsilon_+ & \\
z^+ & \leq & \zeta_+ & \\
\theta, z^-, y^-, y^+, z^+ & \geq & 0. &
\end{array}$$

Remarque : L'absence des termes de stabilisation pour une contrainte se traduit, dans la fonction de pénalité, par des bornes infinies et des pénalités nulles.

5.1.2 Cas d'une fonction de pénalité linéaire à quatre morceaux

Tel que mentionné dans sa définition, la fonction de pénalité à quatre morceaux ne diffère de la fonction de pénalité à cinq morceaux que par l'absence des intervalles de confiance et de pénalisation à gauche du centre de stabilité $\hat{\pi}$. Par conséquent, les formulations des problèmes (SMD_l) et (SMP_l) deviennent :

$$(SMD_l)$$

$$Max \sum_{i=1}^{m}(b_i\pi_i - \varepsilon_{-,i}u_i^- - \varepsilon_{+,i}u_i^+ - \zeta_{+,i}v_i^+) - \sum_{k\in K} n_k\lambda^k$$

$$\begin{aligned}
\sum_{i=1}^{m} a_{ip}\pi_i \;-\; \sum_{k\in K} b_p^k\lambda^k \;&\leq\; c_p, && \forall p \in \Omega \\
-u_i^- \;\leq\; \pi_i \;&\leq\; \delta_{+,i} + u_i^+, && i = 1,\ldots,m \\
\pi_i \;&\leq\; \gamma_{+,i} + v_i^+, && i = 1,\ldots,m \\
u^-,\ \lambda,\ u^+, v^+ \;&\geq\; 0
\end{aligned}$$

$$(SMP_l)$$

$$Min \sum_{p\in\Omega} c_p\theta_p + \sum_{i=1}^{m}(\delta_{+,i}^l y_i^+ + \gamma_{+,i}^l z_i^+)$$

$$\begin{aligned}
\sum_{p\in\Omega} a_{ip}\theta_p - y_i^- + y_i^+ + z_i^+ \;&=\; 1, && i = 1,\ldots,m \\
\sum_{p\in\Omega} b_p^k\theta_p \;&\leq\; n_k, && k = 1,\ldots,|K| \\
y^- \;&\leq\; \varepsilon_- \\
y^+ \;&\leq\; \varepsilon_+ \\
z^+ \;&\leq\; \zeta_+ \\
\theta, y^-, y^+, z^+ \;&\geq\; 0.
\end{aligned}$$

Tableau 5.1 – Limites initiales sur les paramètres de stabilisation

Paramètres	δ_-	γ_-	δ_+	γ_+	ε_-	ζ_-	ε_+	ζ_+
Inf	5	10	10	10	0.0001	1	0.0001	1
Sup	50	100	50	100	0.8000	5	0.8000	5

5.2 Phases d'expérimentation préliminaires

Quoiqu'ayant prouvé une efficacité certaine à plus d'un échelon, les techniques de stabilisation demeurent sujettes à une déficience pratique dès qu'il s'agit d'opérer une projection des paramètres expérimentaux propres à une application connue sur d'autres applications. En l'occurrence, notre travail se veut l'ébauche d'une voie orientée dans ce sens et dont nous tenterons de baliser les contours à mesure que progressera l'analyse en cours.

5.2.1 Limites fixes, paramètres de stabilisation initiaux différents

Cette étape de l'étude a fondamentalement été utile pour tester l'importance des limites sur les paramètres de stabilisation et, surtout, apprécier l'interaction de celles-ci avec les valeurs choisies pour les paramètres de stabilisation initiaux. Voulant lancer la recherche sur des bases "sûres", nous avons retenu les mêmes limites sur les paramètres de stabilisation établies par Ben Amor [1] pour le cas spécifique de son application (Tableau 5.1). Nous avons ensuite résolu les problèmes ainsi stabilisés en adoptant la stratégie hybride. Une autre phase importante consiste à fixer un jeu de paramètres initiaux $\{\zeta_-, \varepsilon_-, \gamma_-, \delta_-, \delta_+, \gamma_+, \varepsilon_+, \zeta_+\}$ spécifique à chaque problème. Plusieurs valeurs sont testées jusqu'à ce que l'optimum du problème stabilisé soit

Tableau 5.2 – Valeurs initiales des paramètres de stabilisation

Par.	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\delta_{\pm}$	10	5	10	5	1	1	1	1	1	1	1	10	1	1
$\gamma_{\pm}$	30	25	30	25	5	5	5	10	10	10	10	30	5	5
$\varepsilon_{\pm}$	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.1	0.1	0.1	0.1	0.3	0.7	0.7
$\zeta_{\pm}$	1	1	1	1	1	1	1	1	1	1	1	1	0.4	0.4

obtenu en un temps CPU raisonnable. Le tableau 5.2 donne la liste des jeux de paramètres utilisés. Nous avons obtenu les résultats du tableau 5.3. **Fr-cpu** désignant le rapport de réduction du temps CPU suite à la stabilisation préliminaire de la génération de colonnes, il semble que les jeux de paramètres initialement utilisés ne produisent pas l'amélioration des temps CPU escomptée. En effet, à partir de l'horizon 4 jours, le temps CPU total frôle les 5 *heures*, ajouté au fait que les problèmes de l'horizon 7 jours ont toujours des difficultés à se résoudre en des temps raisonnables.

Forts de ce dernier constat, nous entreprenons une investigation minutieuse de la problématique, axée entièrement sur le problème 9, lequel est le plus difficile de l'horizon 4 jours, se fixant comme objectif principal l'amélioration de ce temps au mieux. À cet effet, nous recherchons la meilleure combinaison {Paramètres de stabilisation initiaux- Limites sur les paramètres de stabilisation} en opérant de manière alternative des modifications sur les paramètres de stabilisation initiaux et leurs limites. Dans ce qui suit, nous allons expliquer notre approche quant à fixer les valeurs initiales ainsi que les limites des paramètres de stabilisation conformément à certains résultats pratiques.

5.2.2 Choix des limites sur les paramètres de stabilisation

Si important qu'il soit, le choix des paramètres de stabilisation initiaux est loin de constituer une fin en soit. Il est encore plus pertinent d'en contrôler la variation à

Tableau 5.3 – Résultats préliminaires de l'approche de stabilisation

Pb.	MP-cpu (s)	SP-cpu (s)	TL-cpu (s)	Gen	ItBar	ItGen	Fr-cpu(%)
1	17.70	23.00	43.10	2480	5	51	8.58
2	12.80	11.60	26.20	1989	6	37	13.75
3	98.90	60.30	162.20	3996	10	93	10.54
4	61.40	58.40	123.10	3204	11	77	17.19
5	799.40	231.40	1035.80	6827	23	238	13.42
6	999.60	200.50	1204.50	7108	19	236	20.79
7	5304.30	304.50	5595.30	17233	64	644	6.97
8	1859.40	312.90	2177.20	12487	16	423	15.08
9	15880.70	234.60	16122.20	30384	113	1198	8.56
10	2375.60	154.00	2533.60	13550	20	525	1.82
11	22316.10	399.70	22723.90	34183	140	1469	7.34
12	15750.10	329.20	16085.90	26455	192	1089	8.35
13	48344.60	335.50	48691.50	45915	162	2138	5.24
14	17415.80	100.90	15268.30	23784	40	1031	3.50

chaque fois que la stratégie de mise à jour est activée, de sorte à rester fidèle à la vocation même du processus de stabilisation qui est de "restreindre les déplacements inutiles dans l'espace dual". Dans ce sens, nous avons voulu initier une approche plus ou moins fiable pour estimer les limites supérieures (Sup) et inférieures (Inf) sur les valeurs prises par les paramètres de stabilisation.

Connaissant le nombre optimal n_v^* de véhicules fourni par la résolution de la relaxation linéaire du problème *MDVSP*, de même que l'écart entre son optimum Z_m et l'optimum du problème *SDVSP* correspondant Z_s, il est facile d'estimer l'augmentation moyenne des coûts associés aux tournées optimales de la relaxation linéaire du problème *MDVSP* par $\bar{e_r}$ tel que :

$$\bar{e_r} = \frac{Z_m - Z_s}{n_v^*}$$

Nous convenons d'adopter la valeur ainsi estimée comme ordre de grandeur sur l'amplitude des intervalles de confiance et de pénalisation. Aussi, pour le cas particulier

Tableau 5.4 – Limites sur les paramètres de stabilisation testés

Limites	**L0**	**L1**	**L2**	**L3**	**L4**	**L5**	**L6**	**L7**	**L8**
Inf ($\delta_{\pm}$)	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Sup ($\delta_{\pm}$)	1	1	1	1	1	0.8	0.8	0.8	0.8
Inf ($\gamma_{\pm}$)	5	5	10	8	8	5	5	10	8
Sup ($\gamma_{\pm}$)	10	15	15	10	15	10	15	15	10

du problème 9, objet de l'investigation en cours, $\bar{e_r} \approx 7$. Ceci permet d'établir approximativement la borne supérieure sur l'amplitude de γ à 10 et la borne inférieure à 5. Pour δ, celles-ci sont calculées par le rapport

$$\bar{e_r}' = \frac{Z_m - Z_s}{n}$$

représentant la répartition moyenne de l'écart E sur les n tâches. Pour le cas du problème 9, $\bar{e_r}' = 0.4$ donne lieu à une approximation de l'amplitude maximale de δ à 1 et de l'amplitude minimale à 0.5. Néanmoins, cette approche manquerait de rigueur dans un contexte "réel" où la valeur de Z_m est inconnue, tout l'effort étant, en fait, d'en déterminer la valeur en résolvant le problème *MDVSP* lui même. Dans cette optique, il serait peut-être plus judicieux d'estimer $\bar{e_r}$ en passant par des résultats plus faciles à obtenir, comme $\bar{Z}$, l'optimum du problème de partitionnement discuté au chapitre troisième.

Ainsi, remarquant que le nombre optimal de véhicules n_s^* engagés par les tournées du problème *SDVSP* coincide dans tous les cas étudiés avec n_v^* (voir Tableau 5.5), nous pourrons calculer les écarts moyens $\bar{E}$ par la formule :

$$\bar{E} = \frac{\bar{Z} - Z_s}{n_s^*}$$

Par la suite, tout l'intérêt est d'analyser le lien entre $\bar{E}$ et $\bar{e_r}$ relativement à chaque horizon h pour qu'une approximation de $\bar{e_r}$ soit possible sans connaître Z_m. Cette idée peut, en outre, être exploitée ultérieurement pour une révision à la baisse de

Tableau 5.5 – Estimation des écarts moyens

Problème	Z_m	n_v^*	Z_s	$\bar{e}_r$	$\bar{Z}$	n_s^*	$\bar{E}$
1	1254839.24	117	1253697	9.76	1259075	117	45.97
2	1493708.33	140	1491588	15.15	1499549	140	56.86
3	961627.00	87	961026	6.91	966234	87	59.86
4	1083751.57	99	1081809	19.62	1091223	99	95.09
5	619355.44	50	619126	4.59	622812	50	73.72
6	615723.35	50	615529	3.89	619045	50	70.32
7	474810.50	34	474598	6.25	476604	34	59.00
8	580954.72	40	580704	6.27	584447.5	40	93.59
9	436472.00	28	436274	7.07	438284	28	71.79
10	464399.64	30	464337	2.09	465310	30	32.43
11	458586.50	25	458439	5.90	460552	25	84.52
12	472440.00	29	472332	3.72	475205.75	29	99.09
13	402042.00	22	401864	8.09	403437	22	71.50
14	454186.00	25	454116	2.80	455519	25	56.12
15	440150.00	20	440135	0.75	440695	20	28.00
16	441170.00	20	441142	1.40	442060	20	45.90

la valeur de la borne supérieure $\bar{Z}$ et, subséquemment, opérer une réduction plus en profondeur du réseau.

Pour le cas de notre application, il est prématuré de s'avancer définitivement sur la nature de la relation entre $\bar{E}$ et $\bar{e_r}$ vue la taille des échantillons considérés à la fois en termes du nombre de problèmes retenus par horizon ou encore de l'étendue des horizons eux-mêmes. Par contre, pareille approche peut être amplement bénéfique si elle est appréhendée dans les normes de l'inférence statistique.

Tableau 5.6 – Jeux de paramètres de stabilisation initiaux testés

Par.	J0	J1	J2	J3	J4	J5	J6	J7	J8	J9	J10	J11
ζ_-	1	1	1	1	1	1	1	1	1	1	1	1
γ_-	1	1	1	1	1	1	1	1	1	.01	.1	1
ε_-	.1	.1	.1	.1	.1	.1	.1	.1	.1	.1	.1	.1
δ_-	0	0	0	.01	.001	.1	.01	.001	.1	.0001	.001	0
δ_+	.01	.001	.1	.01	.001	.1	0	0	0	.0001	.001	1
ε_+	.1	.1	.1	.1	.1	.1	.1	.1	.1	.1	.1	.1
γ_+	1	1	1	1	1	1	1	1	1	.01	.1	1
ζ_+	1	1	1	1	1	1	1	1	1	1	1	1

5.2.3 Choix des valeurs initiales des paramètres de stabilisation

Se voulant une approche valable quelque soit l'horizon, les valeurs initiales des paramètres de stabilisation ont été choisies de sorte à satisfaire au maximum les particularités des problèmes étudiés. Par conséquent, nous choisissons pour plus grandes valeurs initiales de δ et γ la même valeur 1, estimée plus haut par le biais de $\bar{e_r}'$. Concernant ζ et ε, leurs valeurs initiales sont fixées à 1 et 0.1 respectivement, vérifiant la condition établie antérieurement, voulant que :

$$\zeta + \varepsilon > 1.$$

Lors de l'application, le passage d'un jeu de paramètres à l'autre s'est effectué sur la base de la nature des résultats courants, et leurs nouvelles valeurs reposent beaucoup plus sur l'intuition. Nous avons ainsi mené nos investigations sur un ensemble de 12 jeux de paramètres initiaux, répertoriés dans le tableau 5.6, lesquels sont révisés lors de l'application de la fonction de pénalité à quatre morceaux.

5.2.4 Recherche des caractéristiques "extensibles" des paramètres

L'idée centrale consistant à identifier des paramètres de stabilisation dont les caractéristiques (valeurs initiales et limites) sont extensibles à tous les horizons et, éventuellement, à des problèmes de plus grande taille, l'étude a suivi le cheminement en entonnoir décrit ci-dessous et illustré par la figure 5.1.
Il s'agira d'exploiter l'effet de l'augmentation de la taille du réseau sur le temps CPU pour localiser des caractéristiques "extensibles" des paramètres de stabilisation. L'ensemble L des limites sur les paramètres de stabilisation, de même que l'ensemble J de leurs valeurs initiales se verront réduits à mesure que la taille du problème augmente. Finalement, une seule variante sur les limites des paramètres de stabilisation sera retenue pour tout le reste de l'étude.
Voici l'algorithme de sélection des meilleures limites sur les valeurs des paramètres de stabilisation.

Etape 1 : Générer pour le même horizon de 4 jours, des problèmes de tailles successives 600, 700, 800, 900, 1000 que l'on désignera par p-600, p-700, p-800, p-900, p-1000 respectivement.
Chaque problème stabilisé est désormais défini par le triplet $(j,\ L_i,\ J_k)$ où j est la taille du problème ($j \in \{500,\ 600,\ 700,\ 800,\ 900,\ 1000\}$), L_i est le vecteur des limites sur les paramètres de stabilisation ($L_i \in L,\ i = 0, \ldots, 8$) et J_k est le jeu des valeurs initiales des paramètres de stabilisation ($J_k \in J,\ k = 0, \ldots, 11$).

Etape 2 : Essayer de résoudre les problèmes ainsi générés progressivement et, à chaque taille supérieure, déceler les limites sur les paramètres de stabilisation les moins intéressantes et les éliminer. Ce processus se déroulera comme suit :

- Pour j fixé, démarrer simultanément l'exécution de tous les problèmes stabilisés correspondant aux triplets $(j,\ L_i,\ J_k)$.

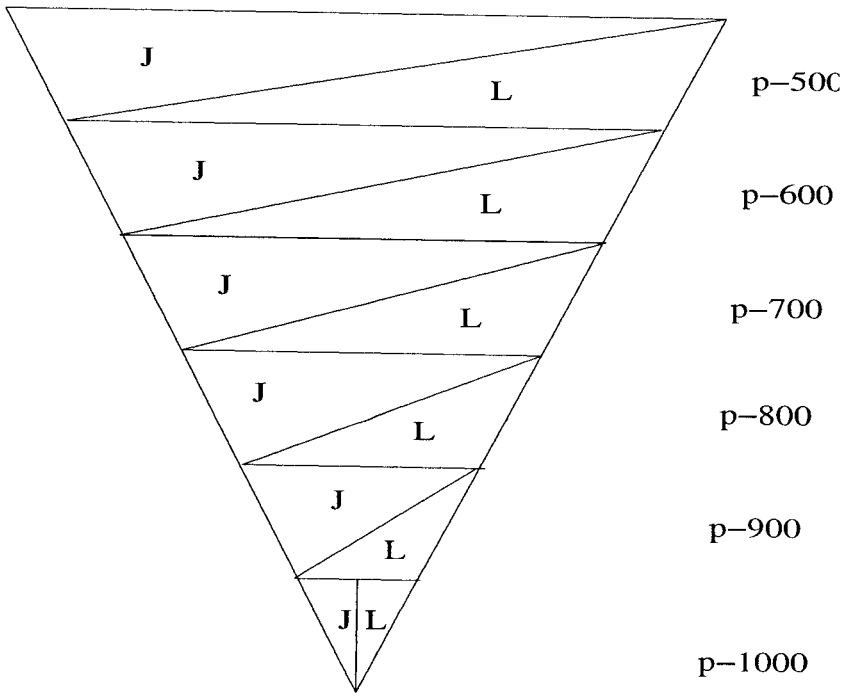

Figure 5.1 – Recherche des caractéristiques "extensibles" des paramètres

- A mesure que le processus de calcul fournit des résultats, repérer le temps CPU de référence T_r, qui n'est autre que le temps CPU le plus faible de la séquence en cours d'exécution.
- Estimer un temps CPU acceptable T_a en multipliant T_r par un coefficient qui sera fixé relativement à un ordre de grandeur intuitif ou encore pratiquement admissible.
- Interrompre l'exécution en cours de tout problème stabilisé dont le temps CPU dépasserait T_a.
- Si le triplet $(j,\ L_i,\ J_k)$ ne se résout à temps pour aucun jeu de paramètres initiaux J_k, la variante L_i est écartée définitivement pour la taille j du problème en cours ainsi que pour toutes les tailles supérieures. Sinon, recommencer la procédure pour la taille j suivante en tenant compte des ajustements précédents.
- Finalement, pour la dernière taille de problème explorée ($j = 1000$), retenir uniquement la variante L_i dont le temps CPU est en moyenne le meilleur et contenant le

Tableau 5.7 – Résultats de base des problèmes générés

Problème	Z_m	n_v^*	Z_s	$\bar{e}_r$	$\bar{Z}$	n_s^*	$\bar{E}$
p-500	436472	28	436274	7.07	438284	28	71.79
p-600	539124	36	538940	5.11	542124	36	88.44
p-700	590333	40	590147	4.65	592646	40	62.48
p-800	644724	42	644657	1.60	647615	42	70.43
p-900	737962	48	737797	3.44	740836	48	63.31
p-1000	803663	48	803601	1.29	807235	51	71.25

jeu de paramètres initiaux qui correspond au temps CPU le plus faible par rapport à tous les problèmes stabilisés de taille j ($j = 1000$ dans notre cas.)

Remarque : Pour cette phase de l'étude, la mise à jour des paramètres de stabilisation a été effectuée par la stratégie dynamique bilatérale dont les performances seront testées aux sections suivantes.

Les résultats préliminaires à la stabilisation sont donnés par le tableau 5.7. L'écart moyen $\bar{e}_r$ a tendance à diminuer à mesure que la taille du réseau augmente alors que $\bar{E}$ semble plus ou moins conserver le même ordre de grandeur.

La mise en œuvre de la procédure de stabilisation sur les relaxations linéaires des problèmes *MDVSP* a fourni les optima en des temps CPU moyens (en secondes) relevés au tableau 5.8. Les cases vides correspondent à l'élimination des limites sur les paramètres de stabilisation. Ainsi, les seules limites à rester intéressantes pour toutes les tailles de réseau sont L0 et L2, alors que toutes les autres limites sont écartées à partir du problème de taille 800. Finalement, L0 et L2 seront les seules limites en ballotage.

En se basant sur les temps CPU minimaux requis pour la résolution de ces problèmes (Tableau 5.9), il en découle que, vraisemblablement, L0 sont les limites sur les para-

Tableau 5.8 – Temps CPU moyens de la stabilisation préliminaire

Problème	p-500	p-600	p-700	p-800	p-900	p-1000
L0	374	878	2687	8229	11467	21029
L1	356	1058	2419			
L2	306	910	2156	6841	11406	22176
L3	363	1068	2464			
L4	354	922	2392			
L5	320	1091	2294			
L6	330	990	2301			
L7	355	890	2268			
L8	334	1031	2908			
Moyennes	344	982	2432	1674	2541	4801

mètres de stabilisation qui sont susceptibles de produire les meilleures performances de l'algorithme de stabilisation pour des problèmes de grande taille.

Un autre aspect intéressant qui mérite d'être mentionné est la variation des temps CPU moyens en fonction de la taille du réseau dont la tendance est très proche d'une fonction polynomiale de degré 3, comme illustré par la figure 5.2.

Bien que la difficulté du problème *MDVSP* causée par l'augmentation de la taille du réseau n'est techniquement pas assimilable à celle résultant de l'allongement de l'horizon, les variations des temps CPU moyens paraissent toutefois similaires pour les deux situations. De ce fait, nous sommes enclins à exploiter les résultats de la présente section dans l'application de la procédure de stabilisation au problème *MDVSP* à horizon long.

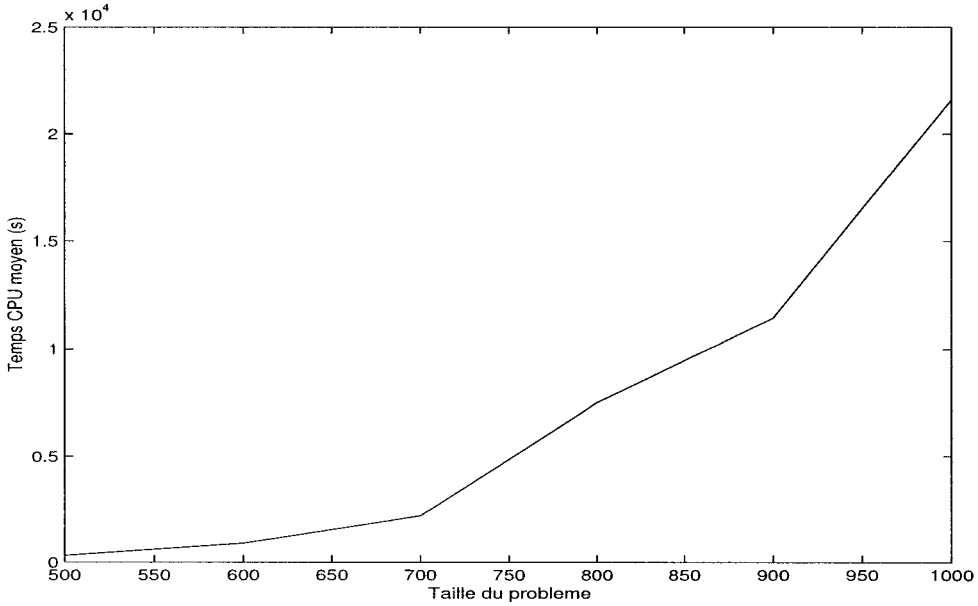

Figure 5.2 – Variation des temps CPU moyens en fonction de la taille du réseau

Tableau 5.9 – Temps CPU minimaux de la stabilisation préliminaire

Problème	p-500	p-600	p-700	p-800	p-900	p-1000
L0	277.30	718.50	1511.80	5659.10	10094.00	16440.90
L1	264.70	751.90	1944.30			
L2	232.80	738.20	1655.60	5948.40	10043.90	17783.90
L3	258.60	754.90	1978.50			
L4	242.80	699.50	1674.50			
L5	223.40	811.30	1666.60			
L6	271.80	771.40	1737.40			
L7	284.10	743.20	1655.30			
L8	335.01	1048.69	2546.75			
Minima	223.40	699.50	1511.80	5659.10	10043.90	16440.90

5.3 Application de la procédure de stabilisation par une fonction linéaire à cinq morceaux

La mise en œuvre de la stabilisation de la procédure de génération de colonnes est désormais entreprise moyennant les principales caractéristiques de stabilisation fixées à la section 5.2. Par ailleurs, l'analyse du comportement de l'algorithme de stabilisation est menée dans des contextes techniques différents, liés notamment à la résolution des problèmes maîtres, à la mise à jour des paramètres de stabilisation ainsi qu'aux valeurs duales initiales.

L'application de la procédure de génération de colonnes stabilisée concerne la relaxation linéaire des mêmes problèmes *MDVSP* choisis au chapitre 3 pour les différents horizons. Chaque problème est résolu pour tous les jeux de paramètres de stabilisation initiaux du tableau 5.6 dont la mise à jour est envisagée dans les seules limites L0 du tableau 5.4. En outre, tous les calculs sont effectués sur la même machine (serveur d'exécution Origin 2000 de SUN Microsystems [Microprocesseurs $400Mhz$]). Pour des fins de fiabilité de l'analyse et de la comparaison des résultats, nous gardons aussi les paramètres de *GENCOL* inchangés en dehors des paramètres de fixation des méthodes à adopter pour la résolution des problèmes maîtres restreints. L'analyse des résultats est effectuée en moyenne avant de nous intéresser aux meilleurs résultats.

5.3.1 Analyse en moyenne des résultats de la stabilisation

Pour notre application, nous effectuons la stabilisation sur chacun des problèmes *MDVSP* en testant, à chaque fois, les 12 jeux de paramètres de stabilisation initiaux J0, J1, ..., J11 du tableau 5.6 sur ce même problème. Ensuite, nous relevons le

temps CPU et le nombre d'itérations *GENCOL* requis pour la résolution de chacun des problèmes stabilisés, avant d'en calculer la moyenne arithmétique sur la base de l'échantillon de valeurs (temps CPU et nombres d'itérations de génération de colonnes) obtenues sur les 12 jeux de paramètres.

L'idée de l'analyse en moyenne tend à omettre momentanément l'effet des paramètres de stabilisation initiaux et accorder plus de crédit à d'autres aspects techniques. Les moyennes sont examinées par rapport à trois aspects principaux : la stratégie de mise à jour des paramètres de stabilisation, le vecteur des valeurs duales initiales et, enfin, la technique de résolution des problèmes maîtres restreints. Pour ce dernier point, le logiciel *GENCOL* offre comme options l'utilisation de l'optimiseur barrière de CPLEX comme alternative à la méthode primale du simplexe lorsque le nombre d'itérations du simplexe se voient trop élevé. Cette option est identifiée par *pb* (**p**rimal-**b**arrière). Nous testons ensuite l'option *dD* qui consiste à résoudre le problème maître par la méthode duale du simplexe. Pour chacune de ces deux options, la procédure de stabilisation est appliquée avec toutes les stratégies de mise à jour présentées au chapitre 4 pour les trois variantes de valeurs initiales des variables duales : $\hat{\pi}_s$, $\hat{\pi}_{sr}$ et $\hat{\pi}_m$ (*cf. Chapitre 3*).

Variantes pb-$\hat{\pi}_s$ et dD-$\hat{\pi}_s$:
Les variantes *pb*-$\hat{\pi}_s$ et *dD*-$\hat{\pi}_s$ reposent, toutes les deux, sur l'initialisation de la solution duale du problème *MDVSP* stabilisé par le vecteur solution optimale du problème *SDVSP* mais adoptent des options différentes quant à la résolution des problèmes maîtres restreints. Ces variantes sont testées avec toutes les stratégies de mise à jour des paramètres de stabilisation et les résultats moyens obtenus sont présentés au tableau B.1 de l'annexe B.

En moyenne, la procédure de stabilisation de la méthode de génération de colonnes appliquée avec la stratégie hybride permet de réduire significativement les temps CPU

pour tous les horizons. Quelque soit l'option retenue pour le contrôle de l'optimisation du problème maître, le facteur de réduction des temps CPU tend à augmenter à mesure que l'horizon progresse, atteignant 1048 pour le problème le plus difficile de l'horizon 6 jours (problème 13). En effet, ce problème qui se résolvait en 258 *heures* avec la procédure de génération de colonnes standard se résout en moyenne en moins de 15 *minutes* avec l'option *pb*. Quant aux problèmes de l'horizon 7 jours, la résolution par la méthode de génération de colonnes standard n'ayant pas été fructueuse (après plus de 5 *mois* de calcul), l'évaluation du facteur de réduction des temps CPU n'est pas possible. Toutefois, la procédure de stabilisation parvient, en moyenne, à assurer une convergence de l'algorithme de génération de colonnes en moins de 1.7 *heures* pour le problème 15 de cet horizon avec l'option *pb* alors que le problème 16 se résout en moins de 11 *minutes* avec l'option *dD*. Parallèlement, nous remarquons qu'en moyenne, le nombre d'itérations de génération de colonnes fluctue de manière non monotone, marquant une réduction maximale d'un rapport de 14 pour le problème 13 sans tenir compte, une fois encore, de l'horizon 7 jours.

Avec la stratégie dynamique bilatérale, la procédure de stabilisation performe, en moyenne, mieux que la procédure hybride, la réduction des temps s'opérant à un rapport maximal de 2194, noté pour le problème 14, lequel se résout en 3.3 *minutes* en moyenne avec l'option *dD*. Le même constat reste valable pour le nombre d'itérations de génération de colonnes dont le facteur de réduction dépasse 38. Pour ce qui est des problèmes de l'horizon 7 jours, le meilleur temps CPU moyen est de 4 *minutes*. Notons que l'écart entre les facteurs de réduction des temps CPU moyens commence à se creuser entre les options *pb* et *dD* à partir de l'horizon 4 jours nonobstant la stratégie de mise à jour.

D'un autre angle, l'évaluation en moyenne des temps CPU révèle que, pour 75% des problèmes, tout horizon confondu, l'option *pb* est supposée fournir des résultats sensiblement meilleurs que l'option *dD* avec la stratégie hybride, un scénario qui

s'inverse pour le nombre d'itérations de génération de colonnes. Parallèlement, les proportions coincident mieux avec la stratégie dynamique bilatérale pour laquelle l'option dD semble être un outil d'accélération parfait de la procédure de génération de colonnes pour 87.5% des problèmes stabilisés. Le rapport entre les facteurs de réduction relevés pour les deux options dépasse 26, tel est le cas des problèmes de l'horizon 6 jours, confirmant a priori la thèse selon laquelle l'option dD aurait un effet accélérateur non négligeable.

Variantes pb-$\hat{\pi}_{sr}$ et dD-$\hat{\pi}_{sr}$:

Ces variantes de l'application de la procédure de stabilisation font appel à $\hat{\pi}_{sr}$ pour initialiser les valeurs des variables duales. Les résultats moyens obtenus sont présentés au tableau B.2 de l'annexe B.

Une première analyse de ces résultats donne, pour la stratégie hybride, un rapport de réduction des temps CPU moyens excédant 1083 pour le problème 14 (horizon 6 jours), ce qui pourrait être considéré comme une amélioration au regard des résultats obtenus pour le même problème avec $\hat{\pi}_s$. La résolution du problème 16 (horizon 7 jours) s'effectue en une demi-heure avec l'option dD, soit près du cinquième du temps CPU moyen obtenu en initialisant les valeurs duales avec $\hat{\pi}_s$. Cependant, l'amélioration due aux nouvelles valeurs initiales des variables duales se trouve vérifiée pour seulement 43.75% des résultats, toute option confondue. Une image presque similaire se dessine pour le nombre d'itérations de génération de colonnes.

Quant à la stratégie dynamique bilatérale, ce qui la caratérise pour $\hat{\pi}_{sr}$ est un facteur de réduction des temps CPU de l'ordre de 2344 atteint pour le problème 14, esquissant une nette amélioration par rapport aux résultats de $\hat{\pi}_s$. S'agissant du nombre d'itérations de génération de colonnes, la tendance se maintient avec un facteur de réduction de plus de 41. En fait, l'initialisation des variables duales avec $\hat{\pi}_s$ produit des facteurs de réduction plus élevés pour 59.38% des résultats comparée à l'initialisation

avec $\hat{\pi}_{sr}$, ce qui laisse la procédure de stabilisation globalement plus performante avec $\hat{\pi}_s$ plutôt que $\hat{\pi}_{sr}$.

A l'échelle de la résolution des problèmes maîtres restreints, une domination de l'option *dD* par rapport à l'option *pb* est visible pour 62.5% des problèmes résolus avec la stratégie hybride. Cette domination est encore renforcée avec la stratégie dynamique bilatérale pour près de 100% des résultats.

Variantes pb-$\hat{\pi}_m$ et dD-$\hat{\pi}_m$:

Aux paragraphes précédents, l'analyse des résultats moyens a été entreprise sur la base des solutions duales optimales $\hat{\pi}_s$ et $\hat{\pi}_{sr}$ du problème *SDVSP*. Disposant de ces vecteurs comme solutions de base, il est clair que toute combinaison convexe de $\hat{\pi}_s$ et $\hat{\pi}_{sr}$ fournit aussi une solution réalisable au dual du problème *SDVSP*, en l'occurrence, le vecteur des moyennes de $\hat{\pi}_s$ et $\hat{\pi}_{sr}$ donné par :

$$\hat{\pi}_m = \frac{\hat{\pi}_s + \hat{\pi}_{sr}}{2}.$$

Utilisant ce nouveau vecteur pour l'initialisation des variables duales, nous performons la stabilisation dont les résultats sont illustrés au tableau B.3 de l'annexe B.

Avec la stratégie hybride, le temps CPU moyen pour la résolution des problèmes de l'horizon 6 jours se ramène à quelques 4 *minutes* avec l'option *pb*, marquant un facteur de réduction de plus de 3634. Les problèmes de l'horizon 7 jours se résolvent en 2.25 *minutes* avec l'option *dD*. Pour cette stratégie, 56.25% des résultats sont meilleurs avec l'option *dD* plutôt qu'avec l'option *pb*. Pour 69% des temps CPU moyens, l'initialisation des valeurs des variables duales produit plus de réduction avec $\hat{\pi}_m$ qu'avec $\hat{\pi}_s$ alors que cette proportion est de 62.5% lorsque comparée aux résultats obtenus avec $\hat{\pi}_{sr}$. Ceci peut être interprété que, globalement, l'initialisation avec $\hat{\pi}_m$ serait plus adéquate pour une plus grande réduction des temps CPU. Une extension de cette idée serait intéressante pour d'autres combinaisons convexe de $\hat{\pi}_s$ et $\hat{\pi}_{sr}$.

Du côté de la stratégie bilatérale, les temps CPU moyens sont encore plus intéressants, se positionnant à moins de 2 *minutes* pour le problème 16 (horizon 7 jours) et à 2.5 *minutes* pour le problème 14. Le meilleur facteur de réduction est de 3829, obtenu pour le problème 13 dont la résolution s'opère en moyenne en 4 *minutes*. A ce niveau de l'application de la procédure de stabilisation, les facteurs de réduction du nombre des itérations de génération de colonnes se trouvent plus prononcés et, pour 81.25% des problèmes, tout horizon confondu, les performances de l'option *dD* dominent celles de l'option *pb*. D'un autre angle, la comparaison des résultats montre que près de 72% des réductions sont plus importantes lorsque l'initialisation des valeurs des variables duales s'effectue avec $\hat{\pi}_m$ plutôt que $\hat{\pi}_s$ et de l'ordre de 91% comparés à ceux de $\hat{\pi}_{sr}$. Pareille analyse met visiblement en exergue l'apport de l'utilisation conjointe des variables duales $\hat{\pi}_m$ avec la stratégie dynamique bilatérale en plus de la méthode duale du simplexe pour contrôler l'optimisation des problèmes maîtres restreints.

Tableau 5.10 – Limites sur les meilleurs valeurs des critères de performance avec la fonction de pénalité linéaire à cinq morceaux

Valeurs duales initiales	Stratégie	Temps CPU (s)		Facteur de réduction	
		Minimum	Maximum	Minimum	Maximum
$\hat{\pi}_s$	Hybride	83.9	2791.6	2.27	1306.84
	d. b.	34.1	780.1	2.18	3238.08
$\hat{\pi}_{sr}$	Hybride	89.0	4382.7	0.76	1354.29
	d. b.	33.5	1733.7	0.87	2825.20
$\hat{\pi}_m$	Hybride	82.1	3371.7	1.01	4340.05
	d. b.	28.4	964.6	1.68	5042.19

5.3.2 Analyse des meilleurs résultats de la stabilisation

Tel que mentionné plus haut, la section 5.3.1 est plutôt consacrée à une analyse globale du comportement de la procédure de stabilisation ignorant l'effet des paramètres de stabilisation initiaux. Dans cette section, nous adoptons une méthodologie

d'analyse similaire mais en mettant l'accent sur l'apport d'un jeu de paramètres plutôt qu'un autre. En fait, les résultats sont analysés par rapport aux meilleurs temps CPU, i.e., au lieu de calculer la moyenne arithmétique sur les 12 jeux de paramètres de stabilisation initiaux. Nous recherchons parmi les 12 valeurs des temps CPU le minimum et nous identifions ensuite le jeu de paramètres qui lui correspond. Outre l'identification des composantes des meilleurs jeux de paramètres initiaux pour chacun des horizons, cette approche peut aussi être adoptée pour situer le meilleur temps CPU par rapport à la moyenne en prévision de l'élaboration d'un plan pratique pour une *résolution simultanée* d'un même problème avec une série de jeux de paramètres. En d'autres termes, si la moyenne des temps CPU produite par une série de jeux

Tableau 5.11 – Meilleurs temps CPU pour les horizons 6 et 7 jours avec la fonction de pénalité linéaire à cinq morceaux

Valeurs duales initiales	Stratégie	Temps CPU (s)							
		Option *pb*				Option *dD*			
		13	14	15	16	13	14	15	16
$\hat{\pi}_s$	Hybride	711	1638	2792	978	837	1643	1912	472
	d.b.	696	161	294	630	780	135	230	491
$\hat{\pi}_{sr}$	Hybride	696	1761	3116	854	686	1764	3808	645
	d.b.	1014	155	1277	751	632	173	1065	606
$\hat{\pi}_m$	Hybride	214	1569	390	150	247	1571	336	119
	d.b.	184	149	143	135	208	141	128	107

de paramètres de stabilisation initiaux est trop proche du meilleur temps CPU correspondant au même problème, alors on pourrait être amené à conclure que tous les jeux de paramètres performent de manière similaire et, par conséquent, il devient moins pertinent de favoriser un jeu par rapport à un autre.

Variantes pb-$\hat{\pi}_s$ et dD-$\hat{\pi}_s$:

En appliquant ces variantes, la stabilisation de la procédure de génération de colonnes

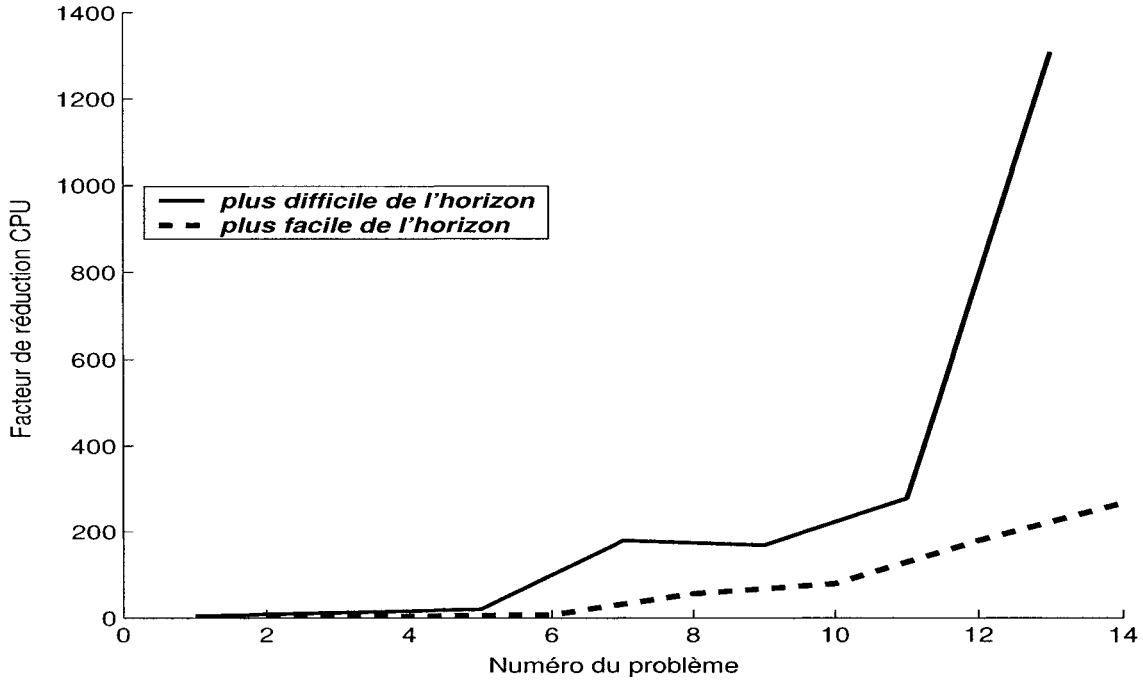

Figure 5.3 – Facteurs de réduction CPU pour la stratégie hybride avec option *pb*

s'opère en des temps CPU dont les facteurs de réduction sont clairement croissants d'un horizon à l'autre, quelque soit la stratégie de mise à jour des paramètres de stabilisation ou la technique de résolution des problèmes maîtres restreints. Cette tendance se voit encore mieux parmi les problèmes les plus difficiles (numéros impairs) ou les problèmes les plus faciles (numéros pairs) pris séparément, comme illustrée sur la figure 5.3, par exemple. En plus, les options *dD* et *pb* se répartissent à moitié les meilleurs facteurs de réduction des temps CPU sans que les écarts ne soient trop profonds. Les résultats sont détaillés au tableau C.2 de l'annexe C.

Avec la stratégie hybride, les problèmes de l'horizon 7 jours se résolvent avec l'option *pb* en des temps CPU représentant plus du double comparativement à l'option *dD*, ce qui donne une fois encore un meilleur avantage à l'option *dD*, les problèmes de l'horizon 7 jours étant classés des plus difficiles. Par ailleurs, tous les problèmes se résolvent en moins de 47 *minutes* avec un meilleur temps CPU de 1.4 *minutes*,

réalisé pour le problème 2. La meilleure réduction des temps CPU connus se retrouve au niveau du problème 13 avec un facteur de 1306 et 907 itérations de génération de colonnes.

Quant à la stratégie dynamique bilatérale, tous les problèmes se résolvent en moins de 13 *minutes*, marquant un temps minimum de 34 *s* pour le problème 2 et un nombre d'itérations de génération de colonnes de 75. Cette fois, l'un des problèmes les plus difficiles (horizon 7 jours) se résout en moins de 4 *minutes* avec un nombre d'itérations de génération de colonnes de 145. La plus grande valeur du facteur de réduction des temps CPU se place à 3238, pour le problème 14, correspondant à un temps CPU de 2.25 *minutes*, réalisé après 205 itérations de génération de colonnes, soit une réduction d'un rapport de 53. Si l'on se base sur ces critères seuls, il semble bien que la stratégie dynamique bilatérale est plus performante que la stratégie hybride bien que, 62.5% seulement des résulats répondent à cette assertion, toute option incluse. Néanmoins, la concentration de cette masse dans la zone de la difficulté (zone commençant à partir de l'horizon 4 jours) donne un poids supplémentaire à ce constat.

Variantes pb-$\hat{\pi}_{sr}$ et dD-$\hat{\pi}_{sr}$:
Les résultats en rapport avec ces variantes sont rapportés au tableau C.3 en annexe C. Généralement, le changement d'option (*pb* ou *dD*) ne semble pas affecter de manière sérieuse les temps CPU sauf quelques exceptions. Cette fois encore, les valeurs des facteurs de réduction des temps CPU gardent la même tendance qu'en moyenne.

Avec la stratégie hybride, les problèmes se résolvent en des temps allant de 1.5 *minutes* à 1.2 *heures*, marquant donc une détérioration des bornes CPU relativement aux résultats obtenus avec $\hat{\pi}_{sr}$. Comparativement au cas de l'initialisation des valeurs des variables duales avec $\hat{\pi}_s$, le problème 13 reste toujours le favori de la liste avec un facteur de réduction de 1354, le nombre des itérations de génération de colonnes étant de 561.

L'usage de la stratégie dynamique bilatérale ramène les bornes des temps CPU à 0.5 *minutes* et 44 *minutes* respectivement, mieux que la stratégie hybride mais moins intéressant que le cas de l'initialisation des valeurs des variables duales avec $\hat{\pi}_s$, d'autant plus que le facteur de réduction des temps CPU correspondant au problème 14 recule à 2520 au lieu de 3238. Parallèlement, le nombre d'itérations de génération de colonnes diminue à 180 pour ce même problème.

Comparativement aux résultats de la section précédente, l'initialisation des valeurs des variables duales avec $\hat{\pi}_{sr}$ n'améliore que le tiers des résultats obtenus avec $\hat{\pi}_s$, l'écart n'étant vraiment significatif que pour le problème 13 de l'horizon 6 jours. Par contre, la stratégie dynamique bilatérale confère un meilleur comportement à l'algorithme pour la résolution des problèmes de la zone facile (problèmes de l'horizon 1 jour à l'horizon 3 jours).

Variantes pb-$\hat{\pi}_m$ et dD-$\hat{\pi}_m$:
L'initialisation des valeurs des variables duales avec $\hat{\pi}_m$ affecte de façon substantielle le comportement de la procédure de stabilisation de la génération des colonnes, comme le montre les résultats du tableau C.4 en annexe C. En effet, le temps de résolution atteint le niveau minimal des 28 s au moment où des problèmes de l'horizon 7 jours se résolvent en moins de 2 *minutes*. Contrairement, le changement d'option (*pb* ou *dD*) ne paraît pas avoir un effet comparable, tout comme ce fut le cas avec $\hat{\pi}_{sr}$.

L'examen des résultats de la stratégie hybride dévoile un facteur de réduction des temps CPU qui dépasse 4340, calculé pour le problème 13, lequel se résout en 3.5 *minutes* au lieu de 11 *jours*, soit une amélioration de 4 fois du temps CPU réalisé en initialisant avec $\hat{\pi}_s$. En outre, tous les problèmes de l'horizon 7 jours atteignent l'optimalité en moins de 6.5 *minutes*, avec le problème 16 qui se résout en moins de 2 *minutes* avec l'option *dD*. Malgré une nette augmentation de la borne

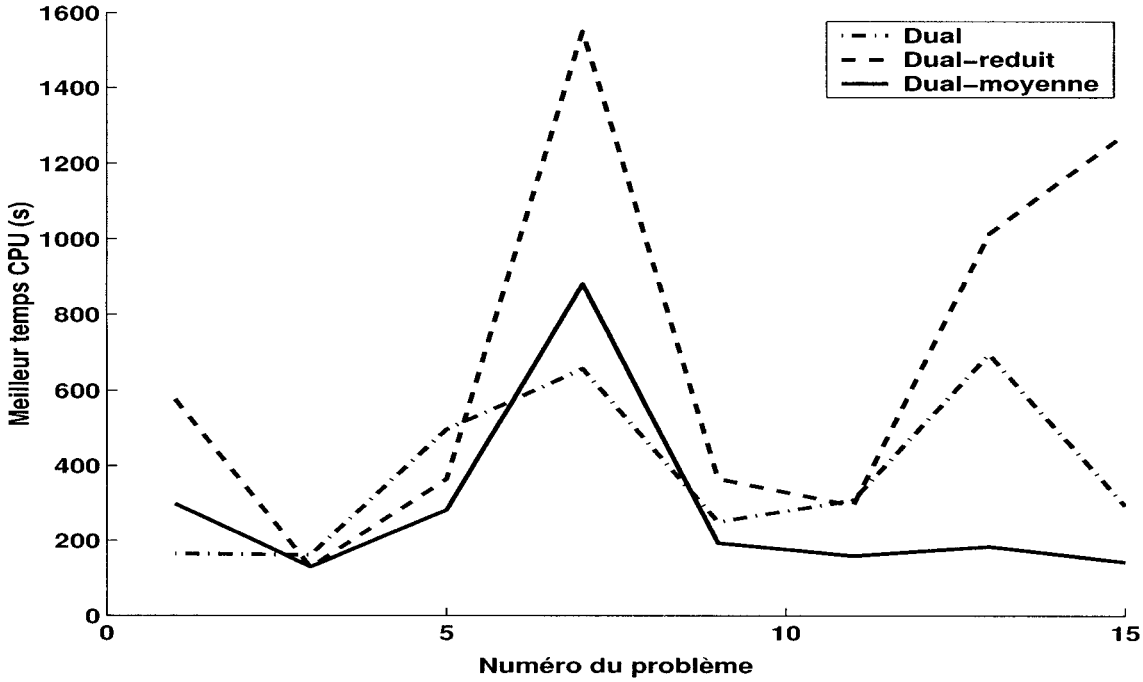

Figure 5.4 – Temps CPU pour la stratégie bilatérale pour différents $\hat{\pi}$

supérieure sur les temps CPU, laquelle dépasse 56 *minutes*, les variantes pb-$\hat{\pi}_m$ et dD-$\hat{\pi}_m$ permettent de résoudre 60% des problèmes stabilisés en des temps meilleurs que les variantes pb-$\hat{\pi}_s$ et dD-$\hat{\pi}_s$ et 78.12% par rapport aux variantes pb-$\hat{\pi}_{sr}$ et dD-$\hat{\pi}_{sr}$.

La stratégie dynamique bilatérale porte le facteur de réduction du temps CPU relatif au problème 13 à plus de 5042. Ceci se produit dans un intervalle de temps CPU borné supérieurement par une valeur de 16 *minutes*. 75% des facteurs de réduction des temps CPU se trouvent améliorés jusqu'à 6 fois en référence à leurs valeurs calculées avec $\hat{\pi}_s$ et 87.5% comparativement aux valeurs obtenues avec $\hat{\pi}_{sr}$. Pratiquement, ces valeurs repères constituent, jusqu'à ce stade de notre analyse, les meilleures performances de la procédure de stabilisation de la génération de colonnes par une fonction de pénalité à cinq morceaux tel qu'illustré par la figure 5.4.

5.4 Application de la procédure de stabilisation par une fonction linéaire à quatre morceaux

Tout au long de la section 5.3, nous avons discuté l'essentiel des résultats fournis par l'application de la procédure de stabilisation de la génération des colonnes par une fonction de pénalité linéaire à cinq morceaux. Bien que les résultats soient encourageants à plus d'un échelon, nous voulons tester une nouvelle approche de la stabilisation à travers une fonction de pénalité linéaire à quatre morceaux. Fidèles aux mêmes notations qu'à la section 5.3, nous menons notre analyse des résultats avec la même méthodologie adoptée plus haut. L'aspect principal qui distingue cette approche de celle discutée aux paragraphes précédents est certainement l'élimination des intervalles de pénalité et de confiance à gauche du centre de stabilité $\hat{\pi}$. Cette caractéristique nous pousse à modifier légèrement les paramètres de stabilisation initiaux de sorte à répondre à cette exigence. Les nouveaux jeux de paramètres utilisés sont présentés au tableau 5.12. Ajouté à cela, d'autres stratégies de mise à jour des paramètres de stabilisation seront de mise, en concordance avec les propriétés de la fonction de pénalisation à quatre morceaux. Les paramètres J12 à J23 sont utilisés en plus de J0 à J11 pour les stratégies hybrides et dynamique bilatérale, et ils sont utilisés seuls avec les stratégies dynamiques unilatérales.

5.4.1 Analyse en moyenne des résultats de la stabilisation

Les conditions d'application de la fonction de pénalité linéaire à quatre morceaux sont similaires à celles explicitées à la section 5.3 exception faite de l'introduction des stratégies dynamiques unilatérales. Tous les résultats ayant trait à cette analyse sont présentés en annexe D.

Si l'on veut situer ces résultats par rapport à ceux de la section 5.3.1, la fonction de pénalité linéaire à quatre morceaux semble, en moyenne, améliorer jusqu'à 26 fois les

Tableau 5.12 – Jeux de paramètres de stabilisation initiaux adaptés

Par.	J12	J13	J14	J15	J16	J17	J18	J19	J20	J21	J22	J23
ζ_-	1	1	1	1	1	1	1	1	1	1	1	1
γ_-	0	0	0	0	0	0	0	0	0	0	0	0
ε_-	.1	.1	.1	.1	.1	.1	.1	.1	.1	.1	.1	.1
δ_-	0	0	0	0	0	0	0	0	0	0	0	0
δ_+	.1	.01	.001	.0001	.1	.01	.001	.0001	.1	.01	.001	.0001
ε_+	.1	.1	.1	.1	.1	.1	.1	.1	.1	.1	.1	.1
γ_+	1	1	1	1	.1	.1	.1	.1	.01	.01	.01	.01
ζ_+	1	1	1	1	1	1	1	1	1	1	1	1

temps CPU obtenus avec la fonction de pénalité linéaire à cinq morceaux pour les problèmes des horizons 0 à 3 jours et perd cet avantage s'agissant des autres horizons, où l'on note un déclin des facteurs de réduction au tiers. Pour la stratégie dynamique bilatérale, les proportions des meilleurs résultats sont plus équilibrées avec, toutefois, une plus grande concentration dans la zone des problèmes faciles pour le cas de la fonction de pénalité linéaire à quatre morceaux. A prime abord, à en juger par ce constat, la fonction de pénalité linéaire à quatre morceaux semble parfaitement appropriée pour des horizons de 0 à 3 jours bien que ses performances ailleurs restent aussi à considérer. Un jugement plus plausible serait certainement possible lors de l'analyse des meilleurs résultats.

Variantes pb-$\hat{\pi}_s$ et dD-$\hat{\pi}_s$:

A première vue, le comportement de l'algorithme de stabilisation ne semble pas s'éloigner de celui observé plus haut quant à la dominance de l'option *pb* par rapport à l'option *dD* pour plus de 81% des cas examinés mais seulement avec la stratégie hybride. Le poids de l'option *pb* marque un déclin à moins de 30% des cas avec les stratégies dynamiques unilatérales pour s'estomper complètement avec la stratégie dynamique bilatérale. Une fois de plus, un jugement en moyenne serait profitable à

l'option *dD* plutôt qu'à l'option *pb*. En outre, la tendance des facteurs de réduction est à la hausse d'un horizon à l'autre, selon que l'on considère séparément les problèmes de numéros pairs et impairs.

Avec la stratégie hybride, les temps CPU moyens se situent entre 29.2 *s* et 4.5 *heures*, une étendue en somme trop large pour apprécier globalement les effets particuliers de l'application de cette stratégie. Toutefois, il est peut être utile de noter que le fateur de réduction des temps CPU atteint un niveau de 1047 pour le problème 13 et, quoique la borne des 4.5 *heures*, correspondant au problème 11, puisse-t-elle sembler excessive, il n'en demeure pas moins que celle-ci réduit en moyenne le temps CPU d'un rapport de 19.

Lorsque la stratégie dynamique bilatérale est mise en œuvre, les temps CPU moyens se réduisent considérablement pour les horizons de la zone difficile pour lesquels le facteur de réduction dépasse le seuil de 2051. Parallèlement, la stratégie hybride garde toujours l'avantage avec des horizons de la zone facile.

L'impact de l'usage de la stratégie dynamique unilatérale simple est perçu pour 62.5% des résultats produits, comparativement à la stratégie hybride et près de 60% comparée à la stratégie dynamique bilatérale, d'autant plus que ces chiffres touchent majoritairement les problèmes des horizons de 3 à 7 jours. Ce même comportement devient plus visible lorsque la stratégie dynamique unilatérale est forcée, avec près de 69% des cas en comparaison aux stratégies hybride ou dynamique bilatérale. Par ailleurs, en forçant la stratégie dynamique unilatérale, une amélioration se produit, en moyenne, pour la moitié des résultats dont la localisation ne peut être restreinte à un groupe d'horizons spécifiés.

Une appréciation globale des résultats des variantes pb-$\hat{\pi}_s$ et dD-$\hat{\pi}_s$ présente les stratégies unilatérales simples comme stratégies de choix pour la résolution des problèmes

des horizons 3 à 7 jours, gardant meilleure la stratégie hybride avec le restant des horizons.

Variantes pb-$\hat{\pi}_{sr}$ et dD-$\hat{\pi}_{sr}$:
L'initialisation des valeurs des variables duales avec $\hat{\pi}_{sr}$ s'avère plus efficace qu'avec $\hat{\pi}_s$ pour 37.5% des problèmes traités lorsque la stratégie hybride est appliquée. Cette proportion frôle 44% avec la stratégie dynamique bilatérale et 50% avec les stratégies dynamiques unilatérales.

Les facteurs de réduction des temps CPU les plus élevés se situent toujours à l'horizon 6 jours, de valeurs respectives 1286, 2462, 2498 et 2471 selon qu'il s'agisse de la stratégie hybride, dynamique bilatérale, dynamique unilatérale simple ou forcée. Conséquemment, il semble fort bien que les stratégies dynamiques sont aptes à produire plus de réduction et, globalement, la stratégie unilatérale simple est, une fois encore, la plus performante même lors de l'initialisation des valeurs des variables duales avec $\hat{\pi}_{sr}$. Toutefois, au regard des proportions recensées en début de section, le recours à $\hat{\pi}_s$ pour l'initialisation reste toujours plus prometteur.

Variantes pb-$\hat{\pi}_m$ et dD-$\hat{\pi}_m$:
Tel qu'escompté, l'initialisation des valeurs des variables duales avec $\hat{\pi}_m$ affecte de manière spectaculaire le comportement de l'algorithme de stabilisation. En effet, les facteurs de réduction des temps CPU sont généralement doublés et, quelques fois même, multipliés par 10 ou 100, tel est le cas des problèmes de l'horizon 7 jours. Les meilleures réductions relevées concernent toujours l'horizon 6 jours, avec des valeurs allant de 2933 à 4553, marquant tout de même un saut important et ce, sans compter l'effet très positif sur les problèmes de l'horizon 7 jours, lesquels se résolvent désormais en des temps CPU moyens allant de moins de 2 *minutes* à 8 *minutes*, quelque soit la stratégie de mise à jour des paramètres de stabilisation [1].

[1]Nous rappelons que les problèmes de l'horizon 7 jours sont restés insolubles par la méthode de génération de colonnes standard malgré le temps de calcul qui leur fut consacré (plus de 5 mois).

Cette particularité des variantes pb-$\hat{\pi}_m$ et dD-$\hat{\pi}_m$ sera mieux mise en relief au cours de l'analyse des meilleurs résultats.

5.4.2 Analyse des meilleurs résultats de la stabilisation

L'analyse des performances de la procédure de stabilisation est, d'abord, entreprise à travers l'examen des limites sur les meilleurs valeurs des critères de performance (Tableau 5.13) ainsi que des meilleurs temps CPU pour les problèmes des horizons 6 et 7 jours, lesquels sont supposés être les plus difficiles (Tableau 5.14). Par la suite, nous procédons à une analyse plus minutieuse basée sur les résultats détaillés disponibles en annexe E. Chacune des variantes examinées ci-dessous sont testées avec toutes les stratégies de mise à jour des paramètres de stabilisation, au nombre de quatre : stratégie hybride, dynamique bilatérale, dynamique unilatérale simple et forcée.

Une première lecture des résultats du tableau 5.13 privilégie l'initialisation des valeurs duales par $\hat{\pi}_s$, quelque soit la stratégie de mise à jour des paramètres de stabilisation adoptée, lorsque l'analyse est effectuée par rapport aux maxima des temps CPU. Par contre, la tendance des maxima des facteurs de réduction favorise mieux l'initialisation des valeurs duales par $\hat{\pi}_m$, le facteur de réduction se multipliant par presque 3 entre $\hat{\pi}_s$ et $\hat{\pi}_m$. Cette tendance est plus apparente sur le tableau 5.14 où l'initialisation des valeurs duales par $\hat{\pi}_m$ produit des résultats meilleurs comparativement à l'initialisation avec $\hat{\pi}_s$, cette option réduisant le temps CPU jusqu'à 9 fois avec l'option *pb*.

A l'échelle des fonctions de pénalité, la confrontation des meilleurs résultats montre une très faible variabilité, quelque soit la stratégie adoptée pour la mise à jour des paramètres de stabilisation, quoi que les performances de la fonction de pénalité linéaire à cinq morceaux semblent globalement meilleures.

L'examen des jeux de paramètres de stabilisation initiaux, relatifs aux meilleurs résultats produits en appliquant une stabilisation par une fonction de pénalité linéaire à quatre morceaux (Annexe E), révèle une parfaite corrélation entre les jeux de paramètres et vecteurs des valeurs initiales des variables duales lorsque les stratégies dynamiques unilatérales sont appliquées. Toutefois, pareil scénario est moins évident quant aux stratégies hybride et dynamique bilatérale bien que l'analyse par composante des jeux de paramètres montre certaines affinités qui conduiraient à une plus grande homogénéité des nuages de points. Ce résultat revêt une importance significative dans la mesure où il ouvre la possibilité de ne considérer qu'un seul jeu de paramètres initiaux pour les calculs futurs, dont la composition sera basée uniquement sur le vecteur des valeurs initiales des variables duales. Par exemple, si la stabilisation s'effectue avec $\hat{\pi}_s$, J22 pourrait être le jeu de paramètres le plus approprié, quelque soit l'horizon des tournées, permettant d'épargner 11 microprocesseurs par horizon, avec toutes les retombées positives sur l'environnement de travail.

Variantes pb-$\hat{\pi}_s$ et dD-$\hat{\pi}_s$:
Les variantes pb-$\hat{\pi}_s$ et dD-$\hat{\pi}_s$ produisent les meilleurs résultats avec la stratégie hybride pour les horizons d'une journée régulière à 2 jours, mais semblent plus performantes avec les stratégies dynamiques pour les horizons de 3 à 7 jours. Ce comportement est d'autant plus visible que le maximum des temps CPU se multiplie par un facteur de 19 entre les deux classes de stratégies (Tableau 5.13). Ceci se confirme mieux pour les problèmes des horizons 6 et 7 jours (Tableau 5.14), lesquels se résolvent en moins de 20 *minutes* lorsque l'une des stratégies dynamiques est appliquée.

Une comparaison des résultats, basée sur la fonction de pénalité utilisée, permet de conclure que la fonction de pénalité linéaire à cinq morceaux permet de réaliser des temps CPU beaucoup plus faibles pour les horizons de 3 à 7 jours lorsque la stratégie hybride est appliquée. Par contre, avec les stratégies dynamiques, il est moins évident de discerner les zones d'influence d'une fonction de pénalité ou d'une autre, la variabilité des résultats n'étant que peu prononcée.

Tableau 5.13 – Limites sur les meilleurs valeurs des critères de performance avec la fonction de pénalité linéaire à quatre morceaux

Valeurs duales initiales	Stratégie	Temps CPU (s)		Facteur de réduction	
		Minimum	Maximum	Minimum	Maximum
$\hat{\pi}_s$	Hybride	28	14404	5	1275
	d.b.	34	1236	2	2181
	d.u.s.	26	760	2	2117
	d.u.f.	29	1217	2	2113
$\hat{\pi}_{sr}$	Hybride	31	19845	1	1508
	d.b.	32	2244	1	2607
	d.u.s.	27	3445	1	2723
	d.u.f.	29	3432	1	2670
$\hat{\pi}_m$	Hybride	26	14652	2	3061
	d.b.	29	1257	1	5010
	d.u.s.	24	2268	1	4914
	d.u.f.	26	2321	1	4870

Variantes pb-$\hat{\pi}_{sr}$ et dD-$\hat{\pi}_{sr}$:

L'initialisation des valeurs des variables duales avec $\hat{\pi}_{sr}$ permet d'améliorer 37.5% des résultats produits en appliquant la stratégie hybride, majoritairement situés dans la zone difficile. Cette amélioration touche près de la moitié des résultats avec les stratégies dynamiques, sans qu'il y ait une zone de concentration apparente. Ceci est parfaitement visible en examinant l'augmentation qui affecte les maxima des facteurs de réduction des temps CPU, lesquels accusent un saut dépassant 600 pour certaines stratégies.

Variantes pb-$\hat{\pi}_m$ et dD-$\hat{\pi}_m$:

L'utilisation de la moyenne de $\hat{\pi}_s$ et $\hat{\pi}_{sr}$ comme estimation des valeurs initiales des variables duales permet à la procédure de stabilisation de produire de meilleurs résultats pour 43.75% des problèmes résolus avec la stratégie hybride. Cette proportion excède 78% avec la stratégie dynamique bilatérale, et frôle 82% avec les stratégies

Tableau 5.14 – Meilleurs temps CPU pour les horizons 6 et 7 jours avec la fonction de pénalité linéaire à quatre morceaux

Valeurs duales initiales	Stratégie	Temps CPU (s)							
		Option *pb*				Option *dD*			
		13	14	15	16	13	14	15	16
$\hat{\pi}_s$	Hybride	728	7851	2261	1074	981	10935	1939	523
	d.b.	651	222	308	1236	767	200	225	522
	d.u.s.	592	229	459	647	760	206	440	527
	d.u.f.	547	229	395	1217	738	207	455	527
$\hat{\pi}_{sr}$	Hybride	616	8812	3459	1007	741	8853	3582	686
	d.b.	1079	167	1491	1080	719	168	1122	767
	d.u.s.	541	160	2173	1121	628	171	2012	784
	d.u.f.	560	164	2249	1130	659	175	1900	767
$\hat{\pi}_m$	Hybride	303	9550	371	173	360	9471	345	139
	d.b.	185	152	151	136	213	140	137	113
	d.u.s.	189	147	222	154	219	145	220	101
	d.u.f.	191	155	190	131	220	147	203	102

dynamiques unilatérales, avec des gains en temps CPU qui dépassent 10 fois pour certains problèmes, comparativement aux cas de l'initialisation avec $\hat{\pi}_s$ ou $\hat{\pi}_{sr}$.

5.5 Conclusion

La qualité des résultats obtenus exprime la satisfaction des objectifs visés par cette étude, surtout pour les problèmes les plus difficiles, tels ceux de l'horizon 6 jours qui se résolvent en moins de 5 *minutes*, enregistrant des rapports de réduction des temps CPU dépassant 5000 *fois* comparativement à la méthode de génération de colonnes standard. Au même moment, les problèmes de l'horizon 7 jours, lesquels n'ont pu être résolus par cette dernière malgré un temps de calcul alloué de plus de 5 mois, se résolvent désormais en moins de 2 minutes.

Mais, s'il y a lieu de synthétiser les nombreux résultats présentés tout au long de ce chapitre en une seule idée, nous dirons : la qualité des résultats escomptés reste fortement tributaire des conditions initiales de l'application.

Convaincus de la teneur d'une telle assertion, nous avons consacré une part non négligeable de notre travail au développement des grandes lignes d'une méthodologie de recherche des *bons* paramètres de stabilisation initiaux. A cet effet, nous sommes parvenus à fixer des paramètres dont l'efficacité se confirme nonobstant certaines spécificités du problème de tournées de véhicules, dont le nombre de tâches, le nombre de dépôts ou encore la longueur de l'horizon. Les principales idées d'approche présentées ont été bénéfiques au contexte de l'application mais leur efficacité serait mieux appréciée une fois traduites sous forme de règles.

Au côté des paramètres de stabilisation, l'initialisation des valeurs des variables duales a aussi constitué l'objet d'une intense investigation dont les résultats ont été plutôt

concluants quant à l'apport de la combinaison convexe de $\hat{\pi}_s$ et $\hat{\pi}_{sr}$. La quantité d'information duale contenue dans la combinaison convexe est l'explication la plus plausible d'un tel résultat, lequel augure des perspectives prometteuses. En effet, cette idée pourrait tout aussi être testée en impliquant d'autres solutions duales, faciles à obtenir, comme, par exemple, celle du problème de partitionnement associé à la solution du *SDVSP*. Une fois les conditions initiales de l'application définies, les stratégies de mise à jour des paramètres de stabilisation sont testées. Globalement, les performances de la stratégie hybride sont prouvées pour les problèmes des horizons s'étendant d'une journée régulière à 2 jours tandis que les stratégies dynamiques sont plus appropriées pour des horizons plus longs.

CONCLUSION

Un jour, tout sera bien, voilà notre espérance
Tout est bien aujourd'hui, voilà l'illusion
Voltaire

Une évaluation objective des résultats reste tributaire des objectifs assignés à l'étude. L'objectif déclaré au départ est bien la résolution du problème de tournées de véhicules multi-dépôts à horizon long en des temps CPU raisonnables. Néanmoins, la progression de la recherche a forcé l'orientation du courant vers une étude numérique d'une approche de stabilisation proximale, une enveloppe en somme logique de l'objectif initial. Ladite recherche s'est développée selon l'enchaînement suivant.

Un premier chapitre a été consacré à la compréhension du caractère spécifique des problèmes de tournées de véhicules multi-dépôts à horizon long. Pour cela, nous avons mis au point un algorithme d'extension de l'horizon des tournées de véhicules, nous avons généré aléatoirement des instances de problèmes test pour des horizons de temps allant d'une journée régulière à 7 jours et nous avons analysé les structures des réseaux qui leurs sont associés. Comme, à ce stade, rien ne pouvait justifier l'aspect difficile du problème traité, le deuxième chapitre a porté sur la résolution de ces problèmes par l'algorithme de génération de colonnes *standard*. Les résultats obtenus ont permis de mettre en exergue la dégénérescence primale à travers l'effet de queue de la génération de colonnes, laquelle devient plus prononcée à mesure que l'horizon s'allonge.

Le troisième chapitre a été axé sur l'élaboration d'une approche heuristique qui réalise deux objectifs essentiels : la réduction de la taille des réseaux associés aux problèmes en adoptant un algorithme basé sur le principe de la fixation des variables, et la

recherche d'une bonne estimation des points duaux initiaux requis à l'implémentation de la procédure de stabilisation. Cette étape a été décisive dans la mesure où certains problèmes se sont vus réduits jusqu'aux trois quarts de leurs tailles initiales, un résultat dont la portée est encore plus grande lorsque les problèmes concernés sont ceux des horizons avancés (5, 6 et 7 jous). En outre, nous avons pu, grâce à une extension de l'algorithme de base, vérifier qu'il y a possibilité d'opérer plus de réduction du réseau. Cette dernière a également été une ouverture sur la diversification des solutions duales initiales en considérant les combinaisons convexes des deux solutions duales de base obtenues en résolvant le *SDVSP* associé à chacun des *MDVSP*. Cette propriété peut tout aussi être étendue en remarquant que les options *opt* et *netopt* de *CPLEX*, utilisées pour la résolution de *SDVSP*, fournissent des solutions primales et duales différentes. La diversité des solutions primales permet d'enrichir les problèmes de partitionnement avec plus de colonnes et, par conséquent, réduire la borne supérieure de *MDVSP* et renforcer la réduction des réseaux. Par contre, les solutions duales seront un excellent moyen d'élargir l'éventail des estimations des points duaux initiaux.

Au quatrième chapitre, nous avons présenté l'algorithme de la méthode proximale adoptée pour la stabilisation du problème dual, les fonctions de pénalité par morceaux utilisées ainsi que les stratégies testées pour la mise à jour des paramètres de stabilisation. L'effort a été centré sur les stratégies dynamiques. Ce qui caractérise ces stratégies est le rapport de réduction ou d'élargissement de l'intervalle de confiance ou de la pénalité, lequel a été fixé à 2. Bien que ce choix soit basé sur des résultats antérieurs, il serait intéressant d'explorer des stratégies utilisant d'autres valeurs.

Le dernier chapitre constitue le noyau de l'application projetée par la présente étude. Nous avons réussi, dans une certaine mesure, à proposer une ébauche pour une méthodologie d'approche laquelle permet de choisir les conditions initiales de la stabilisation nonobstant l'horizon des tournées ou la taille des réseaux. A cet effet, nous sommes

parvenus à définir des limites sur les paramètres de stabilisation valides quelque soit l'horizon considéré et nous avons aussi esquissé les grandes lignes d'un algorithme permettant d'estimer les paramètres de stabilisation initiaux sur la base de résultats faciles à obtenir, tels que les bornes supérieure et inférieure de l'optimum de *MDVSP*. La démarche suggérée était plutôt tronquée, la taille des échantillons en jeu ne satisfaisant pas aux normes statistiques. Par la suite, les conditions propices à l'implémentation de la stabilisation réunies, nous avons mené l'application sur une série de 12 jeux de paramètres de stabilisation initiaux, dérivés principalement de la méthodologie d'approche citée antérieurement. Plusieurs zones d'influence ont été investiguées, en utilisant à chaque fois tous ces paramètres.

La première zone concerne le vecteur des valeurs initiales des variables duales. Cette zone a été particulièrement attrayante tout au long de l'étude en raison de la grande variabilité des résultats, une confirmation de plus que l'information duale reste d'une grande utilité pour résoudre efficacement le primal. Voulant explorer au mieux les résultats recueillis lors de la réduction du réseau, nous avons d'abord testé les solutions optimales duales obtenues en résolvant le *SDVSP* de la première et de la seconde réduction du réseau, partant du constat que l'écart entre la solution optimale de *MDVSP* et celle de *SDVSP* se réduisait à mesure que l'horizon s'allongeait. Toutefois, ceci n'expliquait pas la dégradation des performances de la procédure de stabilisation d'une variante à l'autre. Cette situation nous a conduit à s'interroger sur le comportement probable de la procédure si une combinaison convexe de ces solutions duales venait à être envisagée. Pour le problème de référence de l'étude préliminaire, les résultats ne furent concluants que pour les valeurs voisines de 0.5 du coefficient de la combinaison convexe. Ceci nous amena à anticiper les résultats en ne retenant que cette valeur du coefficient pour l'estimation d'un troisième vecteur des valeurs initiales des variables duales. Avec cette nouvelle estimation, les performances de l'algorithme de stabilisation se vîrent améliorées dans des rapports allant de 10 jusqu'à 100, comparativement à $\hat{\pi}_s$ et $\hat{\pi}_{sr}$.

Un deuxième aspect dont l'impact a été perceptible fut l'approche de résolution des problèmes maîtres restreints. Cet aspect a été inclus pour mesurer l'effet du changement de la technique de résolution des problèmes maîtres restreints sur l'accélération de la procédure de génération de colonnes. De manière globale, la méthode duale du simplexe performe mieux que l'approche primal-barrière pour près des 2/3 des problèmes résolus.

Enfin, les stratégies de mise à jour des paramètres de stabilisation restent au centre de l'influence sur la qualité des résultats. Ce qui ressort de l'étude est la dominance de la stratégie hybride pour les problèmes des horizons de courte durée (1 à 3 jours) au moment où les stratégies dynamiques produisent des résultats manifestement impressionnants lorsque les problèmes en jeu font partie des horizons de longue durée (4 à 7 jours). Ce comportement s'illustre de manière accentuée à travers l'amplitude des coefficients de réduction des temps CPU qui frôlent 6000 pour les problèmes de l'horizon 6 jours, au côté de temps CPU relatifs aux problèmes de l'horizon 7 jours se situant au deçà de 2 min. Bien que les résultats de la présente étude soient édifiants

Le lendemain s'instruit aux leçons de la veille
Publius Syrus (I^{er} siècle Av.JC)

à plus d'un égard, une sensation d'inachevé se profile toujours à l'horizon. Plusieurs interrogations motivent cette sensation : Pourquoi $\hat{\pi}_m$ est plus efficace que $\hat{\pi}_s$ et $\hat{\pi}_{sr}$? Quel comportement adopterait le processus de stabilisation si l'on envisageait l'initialisation avec d'autres solutions duales ou encore des combinaisons convexes impliquant plus de deux vecteurs ? D'autres stratégies de mise à jour des paramètres de stabilisation sont possibles, comment réagirait l'algorithme de stabilisation à leur implémentation ? Autant de questions qui augurent de la fertilité du champ d'investigation et ouvrent l'horizon à des reflexions, et sur le plan théorique, et sur le plan pratique.

Bibliographie

[1] BEN AMOR, H. (2002). Stabilisation de l'algorithme de génération de colonnes. *Thèse de doctorat*, Département de Mathématiques et de Génie Industriel, Ecole Polytechnique de Montréal, Canada.

[2] BEN AMOR, H. et DESROSIERS, J. (2003). A proximal trust-region algorithm for column generation stabilization. *Cahiers du GERAD* G-2003-43, HEC-Montréal, Montréal, Canada.

[3] BIANCO, L., MINGOZZI, A. et RICCIARDELLI, S. (1994). A set partitionning approach to the Multiple Depot Vehicle Scheduling problem. *Optimization Methods and Software*, 3, 163-194.

[4] BORNDÖRFER, R., GRÖTSCHEL, M. et LÖBEL, A. (1995). Alcuin's transportation problems and integer programming. Rapport SC 95-27, Konrad-Zuse-Zentrum Berlin. Disponible sur WWW au www.zib.de.

[5] BUSSIECK, M., WINTER, T. et ZIMMERMANN, U.T. (1997). Discrete optimization in public rail transport. Dans liebling, T.M. and de Werra, D. (Eds, *Mathematical Programming*, 415-444. Elsevier Science B.V.

[6] CARPANETO, G., DELL'AMICO, M., FISCHETTI, M. et TOTH, P. (1989). A branch and bound algorithm for the multiple vehicle scheduling problem. *Networks*, 19, 531-548.

[7] DADUNA, J.R. et PAIXÃO, J.M.P. (1995). Vehicle scheduling for public mass transit- an overview. Dans DADUNA, BRANCO and PAIXÃO.

[8] DANTZIG, G.B. et WOLFE, P. (1960). Decomposition principle for linear programs. *Operations Research*, 8, 101-111.

[9] DESROSIERS, J., DUMAS, Y., SOLOMON, M.M. et SOUMIS, F. (1995). Time constrained routing and scheduling. M.O. Ball *et al.* (eds.), *Handbooks in Operations Research and Management Science, Network Routing*, Vol 8, 35-139, Elsevier, Ville.

[10] DU MERLE, O., VILLENEUVE, D., DESROSIERS, J. et HANSEN, P. (1998). Stabilized column generation. *Discrete Mathematics*, 194, 229-237.

[11] GILMORE, P.C. et GOMORY, R.E. (1961). A linear programming approach to the cutting stock problem. *Operations Research*, 11, 849-859.

[12] GULER, O. (1991). On the convergence of the proximal point algorithm for convex minimization. *SIAM Journal on Control and Optimization*, 29, 403-419.

[13] HADJAR, A., MARCOTTE, O. et SOUMIS, F. (2001). A branch-and-cut algorithm for the multiple-depot vehicle scheduling problem. *Cahiers du GERAD* G-2001-25, HEC-Montréal, Montréal, Canada.

[14] HIRIART-URRUTY, J., LEMARÉCHAL, C. (1991). Convex analysis and minimization algorithms II : advanced theory and bundle methods. *A Series of Comprehensive Studies in Mathematics*, Springer, New-York.

[15] ILOG. ILOG CPLEX 6.6 : User's manual. ILOG, Inc.

[16] KELLEY, J.E.Jr. (1960). The cutting-plane method for solving convex programs. *SIAM Journal on optimization*, 8, 703-712.

[17] KIM, S., CHANG, K-N. et LEE, J-Y. (1995). A descent method with linear programming sub-problems for nondifferentiable convex optimization. *Mathematical Programming*, 71, 17-28.

[18] KIWIEL, K.C. (1989). A survey of bundle methods for nondifferentiable optimization. Dans IRI, M. et TANABE, K. (Eds), *Mathematical Programming : Recent Developments and Applications*, KTT/Kluwer, Tokyo, 263-282.

[19] LASDON, L.S. (1970). Optimization theory for large systems. Macmillan. New-York.

[20] LAVIGNE, J. (1996). Le problème de tournées de véhicules avec fenêtres de temps et dépôts multiples. *Mémoire de maîtrise*, Département de Mathématiques et de Génie Industriel, Ecole Polytechnique de Montréal, Canada.

[21] LEMAIRE, B. (1989). The proximal algorithm. Dans PENOT, J.P. (Eds), *International Series of Numerical Mathematics*, 73-87.

[22] LEMARÉCHAL, C. (1974). An algorithm for minimizing convex functions. Dans ROSENFELD, J.L. (Eds), *Proceedings IFIP'74 Congress,* North-Holland, Amsterdam, 552-556.

[23] LÖBEL, A. (1998). Vehicle scheduling in public transit and lagrangean pricing. *Management Science*, 44, 1637-1649.

[24] MARSTEN, R.E., HOGAN, W.W., BLANKENSHIP, J.W. (1975). The Boxstep method for large-scale optimization. *Operations Research*, 23, 389-405.

[25] MARSTEN, R.E. (1975). The use of Boxstep method in disrete optimization. *Mathematical Programming Study*, 3, 127-144.

[26] MINOUX, M. (1983). Programmation mathématique : Théorie et algorithmes. Dunod, Paris.

[27] NEAME, P. (1999) Nonsmooth methods in integer programming. *Ph.D. Dissertation,* University of Melbourne, Australie.

[28] RIBEIRO, C.C. et SOUMIS, F. (1994). A column generation approach to the multiple-depot vehicle scheduling problem. *Operations Research*, 42 (1), 41-52.

[29] ROCKAFELLAR, R.T. (1976). Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14, 877-898.

[30] SCHRAMM H. et ZOWE, J. (1998). A version of the bundle idea for minimizing a nonsmooth function : conceptual idea, convergence analysis, numerical results. *SIAM Journal on optimization*, 2, 121-152.

[31] VIGNAC B. (2003). Accélération de la convergence de l'algorithme de génération de colonnes. *Rapport de stage*, Université Bordeaux 1, Bordeaux, France.

[32] WOLFE, P. (1975). A method of conjugate gradients for minimizing nondifferentiable convex functions. *Mathematical Programming*, 3, 145-173.

Annexe A : Résultats de l'application de GENCOL standard

Chaque tableau de résultats correspond à un horizon particulier, les problèmes y étant identifiés par $h.i$ où h réfère à l'horizon ($h = 0, \ldots, 7$ *jours*) et i au numéro du problème ($i = 1, \ldots, 5$ *ou* 10).

Tableau A.1 – Résultats de l'application de la méthode de génération de colonnes

Horizon	Pb.	ArcRes	ItGen	NbPCC	Col	TL-cpu (s)
Journée régulière	0.1	76449	151	450	12399	367.1
	0.2	74696	157	468	10924	321.1
	0.3	72182	101	300	8635	190.6
	0.4	75737	188	561	12021	392.9
	0.5	74713	174	519	11590	353.3
	0.6	76292	2117	630	12493	502.6
	0.7	76461	195	582	12805	450.6
	0.8	73683	185	552	11673	359.6
	0.9	73831	204	609	12400	405.9
	0.10	74503	159	474	11456	338.9
Journée complète	1.1	93410	344	1029	18751	1189.3
	1.2	89576	360	1077	17619	1156.7
	1.3	91372	312	933	16069	958.2
	1.4	92275	407	1218	21841	1496.3
	1.5	92638	459	1374	22037	1503.1
	1.6	93667	372	1113	20903	1249.9
	1.7	92202	249	744	14813	716.0
	1.8	91345	385	1152	18148	1267.2
	1.9	92818	479	1434	21481	1539.4
	1.10	92706	490	1367	20834	1493.6

Tableau A.2 – Résultats de l'application de la méthode de génération de colonnes (suite)

Horizon	Pb.	ArcRes	ItGen	NbPCC	Col	TL-cpu (s)
2 jours	2.1	107824	1106	3315	36662	5040.3
	2.2	108164	795	2382	31576	3519.8
	2.3	107910	841	2520	31603	3477.0
	2.4	107918	1195	3582	39739	4038.4
	2.5	107569	1427	4278	43898	4142.4
	2.6	107014	11129	3384	35652	7721.2
	2.7	107864	1029	3084	34809	2933.5
	2.8	107880	1267	3798	38452	3999.9
	2.9	107978	920	2757	32334	2804.1
	2.10	108089	1000	2997	33037	4473.1
3 jours	3.1	112999	2862	8583	78819	28704.5
	3.2	113476	5008	15021	128009	67811.0
	3.3	113729	6590	19767	172577	80266.8
	3.4	113369	1606	4815	43592	14435.5
	3.5	113275	2092	6273	55109	29140.2
4 jours	4.1	116056	7637	21908	183058	139226.5
	4.2	116191	7989	24560	172665	169402.9
	4.3	116408	8134	24399	193573	142990.9
	4.4	116391	7248	21741	182190	138916.1
	4.5	116236	8322	24963	202413	188419.6
5 jours	5.1	117964	21512	64533	494462	560317.5
	5.2	117636	9716.0	29145.0	231124.0	192641.3
	5.3	117999	9980.0	29937.0	233038.0	309701.3
	5.4	117908	26930	80787	611460	557321.5
	5.5	117719	43009	129024	916663	958652.8
6 jours	6.1	120023	17747	53238	398917	928771.3
	6.2	119760	10920	32757	268210	436493.7

Annexe B : Moyennes des critères de performance de la stabilisation par une fonction de pénalité linéaire à cinq morceaux

Chaque tableau correspond à une stratégie de mise à jour à part et comprend deux colonnes principales correspondant aux critères d'évaluation de la procédure de stabilisation : **Temps CPU** et **Nombre d'itérations *GENCOL***. Sous chacune des colonnes principales, il y a quatre sous-colonnes :

- **pb-$\hat{\pi}$.** et **dD-$\hat{\pi}$.** qui correspondent aux valeurs du critère obtenues par les variantes de la stabilisation,
- **rr..** donne le facteur de réduction du critère d'évaluation calculé comme rapport de la valeur critère obtenue par la résolution du problème *MDVSP* par la procédure de génération de colonnes standard sur la valeur du même critère obtenue avec la procédure de stabilisation.

  La troisième lettre réfère à l'option de résolution des problèmes maîtres restreints (p : *pb* et d : *dD*), tandis que la quatrième lettre symbolise le critère concerné (t : *temps CPU* et c : *itérations de génération de colonnes*).

Tableau B.1 – Valeurs pour une initialisation duale avec $\hat{\pi}_s$

Strat.	Pb.	TL-cpu *MDVSP* stabilisé (s)				Itérations *GENCOL*			
		pb-$\hat{\pi}_s$	rrpt	Dd-$\hat{\pi}_s$	rrdt	pb-$\hat{\pi}_s$	rrpc	dD-$\hat{\pi}_s$	rrdc
Hybride	1	217	7	197	8	264	2	238	5
	2	429	2	430	2	540	0	527	0
	3	507	15	528	15	403	3	407	3
	4	2010	3	2325	2	1495	1	1582	1
	5	589	136	704	114	840	8	772	9
	6	2443	6	2621	6	1869	1	1832	1
	7	3537	53	3052	62	2444	3	1980	4
	8	3384	41	4501	31	1678	4	1882	4
	9	5986	52	6511	48	4820	2	4592	2
	10	2951	65	3204	60	2961	3	2143	5
	11	886	1049	1021	909	1758	10	1236	14
	12	5819	75	5840	75	4500	2	4500	2
	13	28647	∞	9575	∞	38735	∞	10774	∞
	14	5900	∞	659	∞	6562	∞	864	∞
d.b.	1	183	8	171	9	231	2	219	2
	2	453	2	443	2	573	0	552	0
	3	555	14	549	14	511	2	501	2
	4	308	19	306	19	248	3	256	3
	5	1242	65	910	88	1807	4	1144	6
	6	320	45	317	46	291	6	291	6
	7	417	452	294	640	615	14	376	22
	8	268	519	285	487	1311	6	1140	6
	9	367	844	383	808	512	20	415	24
	10	446	432	396	487	666	15	471	21
	11	1293	718	833	1115	2412	7	989	18
	12	220	1987	199	2195	329	33	285	38
	13	337	∞	255	∞	168	∞	158	∞
	14	6222	∞	576	∞	8206	∞	866	∞

Tableau B.2 – Valeurs pour une initialisation duale avec $\hat{\pi}_{sr}$

Strat.	Pb.	TL-cpu *MDVSP* stabilisé (s)				Itérations *GENCOL*			
		pb-$\hat{\pi}_{sr}$	rrpt	Dd-$\hat{\pi}_{sr}$	rrdt	pb-$\hat{\pi}_{sr}$	rrpc	dD-$\hat{\pi}_{sr}$	rrdc
Hybride	1	220	7	213	7	241	2	228	2
	2	741	1	696	1	480	1	484	1
	3	858	9	966	8	361	3	370	3
	4	649	9	748	8	278	3	269	3
	5	1289	62	990	81	1218	5	1142	6
	6	378	38	441	33	406	4	444	4
	7	349	540	391	482	551	15	492	17
	8	4904	28	4684	30	3361	2	3097	2
	9	7087	44	7282	43	6115	2	5717	2
	10	6757	29	4241	45	6545	1	3678	3
	11	857	1083	940	988	1762	10	1198	15
	12	5871	74	5850	75	4591	2	4591	2
	13	11194	∞	4165	∞	22829	∞	5108	∞
	14	1219	∞	1050	∞	2429	∞	1502	∞
d.b.	1	155	10	140	11	188	3	174	3
	2	439	2	430	2	556	0	533	0
	3	424	18	406	19	369	3	358	3
	4	334	17	342	17	284	3	278	3
	5	2126	38	1770	45	2407	3	1945	3
	6	316	46	326	44	319	5	305	5
	7	552	342	521	362	1276	7	971	9
	8	274	506	291	477	360	20	356	20
	9	360	861	332	934	425	24	331	30
	10	2009	96	1334	144	1198	8	1135	9
	11	1783	521	915	1015	4683	4	1219	15
	12	186	2344	188	2319	297	37	265	41
	13	2941	∞	1360	∞	5849	∞	1353	∞
	14	974	∞	825	∞	2622	∞	1397	∞

Tableau B.3 – Valeurs pour une initialisation duale avec $\hat{\pi}_{m1}$

Strat.	Pb.	TL-cpu *MDVSP* stabilisé (s)				Itérations *GENCOL*			
		pb-$\hat{\pi}_{m1}$	rrpt	Dd-$\hat{\pi}_{m1}$	rrdt	pb-$\hat{\pi}_{m1}$	rrpc	dD-$\hat{\pi}_{m1}$	rrdc
Hybride	1	174	9	160	10	198	2	176	3
	2	716	1	666	1	481	1	480	1
	3	267	29	269	29	219	5	214	5
	4	1171	5	1517	4	732	1	937	1
	5	1041	77	787	102	830	8	715	9
	6	1688	9	1607	9	1115	1	892	2
	7	2974	63	1596	118	2059	4	885	9
	8	4742	29	4833	29	3089	2	3190	2
	9	6709	46	6747	46	5479	2	5244	2
	10	5138	37	4773	40	2354	4	3280	3
	11	256	3634	313	2971	583	30	524	34
	12	6327	69	6338	69	5098	2	5098	2
	13	953	∞	402	∞	1791	∞	475	∞
	14	181	∞	136	∞	396	∞	233	∞
d.b.	1	144	11	143	11	192	3	191	3
	2	446	2	459	2	565	0	576	0
	3	324	24	305	25	308	4	291	4
	4	273	21	260	22	249	3	231	4
	5	1779	45	1328	60	1794	4	1418	5
	6	208	69	209	69	190	8	185	9
	7	246	766	235	802	430	19	368	23
	8	188	739	214	648	261	28	262	28
	9	174	1780	176	1761	223	45	186	54
	10	1281	150	970	199	2327	4	2107	5
	11	313	2964	243	3830	706	25	375	47
	12	175	2490	153	2859	362	30	294	37
	13	164	∞	146	∞	126	∞	125	∞
	14	185	∞	119	∞	459	∞	212	∞

Annexe C : Meilleurs critères de performance de la stabilisation par une fonction de pénalité linéaire à cinq morceaux

Tableau C.1 – Jeux de paramètres de stabilisation initiaux des meilleurs temps CPU

Pb.	Option pb						Option dD					
	Hybride			d.b.			Hybride			d.b.		
	$\hat{\pi}_m$	$\hat{\pi}_s$	$\hat{\pi}_{sr}$	$\hat{\pi}_m$	$\hat{\pi}_s$	$\hat{\pi}_{sr}$	$\hat{\pi}_m$	$\hat{\pi}_s$	$\hat{\pi}_{sr}$	$\hat{\pi}_m$	$\hat{\pi}_s$	$\hat{\pi}_{sr}$
1	9	9	9	8	0	9	9	10	9	8	4	5
2	9	9	9	2	0	2	9	9	9	2	2	2
3	6	9	5	0	11	11	10	9	5	0	5	2
4	10	9	10	1	1	4	10	9	10	2	3	12
5	9	9	2	2	0	11	9	9	1	9	3	3
6	11	10	11	0	11	2	5	10	6	2	2	2
7	4	10	3	8	0	6	4	10	11	8	2	8
8	11	1	10	2	1	11	3	1	9	10	1	11
9	10	10	10	3	8	11	11	3	3	1	5	11
10	11	11	10	8	10	5	10	11	10	8	10	6
11	10	11	10	6	5	3	10	11	10	3	1	5
12	0	11	9	8	3	8	9	11	10	8	9	8
13	10	5	11	0	4	11	10	8	8	11	1	11
14	10	10	10	8	8	9	10	10	10	0	8	3
15	5	5	4	6	5	7	9	9	9	5	10	9
16	10	6	7	9	6	6	2	11	11	3	11	11

Tableau C.2 – Valeurs pour une initialisation duale avec $\hat{\pi}_s$

Strat.	Pb.	TL-cpu *MDVSP* stabilisé (s)				Itérations *GENCOL*			
		pb-$\hat{\pi}_s$	rrpt	Dd-$\hat{\pi}_s$	rrdt	pb-$\hat{\pi}_s$	rrpc	dD-$\hat{\pi}_s$	rrdc
Hybride	1	119	4	107	5	172	1	139	2
	2	84	2	84	2	118	1	118	1
	3	129	12	108	14	113	4	106	5
	4	214	3	195	4	223	1	209	1
	5	391	20	374	21	272	4	256	4
	6	835	7	1050	6	315	3	325	3
	7	449	179	516	156	535	12	520	13
	8	260	56	292	50	602	3	620	3
	9	1120	168	1402	134	769	11	710	12
	10	1765	79	1986	70	276	26	274	26
	11	1116	277	1132	274	650	15	614	16
	12	1079	179	1263	153	695	14	735	13
	13	711	1307	837	1110	907	20	662	27
	14	1638	267	1643	266	895	12	895	12
	15	2792	∞	1912	∞	3957	∞	1694	∞
	16	978	∞	472	∞	1562	∞	537	∞
d.b.	1	165	3	182	3	325	1	364	1
	2	34	6	34	6	75	1	75	1
	3	161	10	148	10	202	2	189	3
	4	329	2	325	2	401	1	387	1
	5	497	16	498	16	450	3	453	2
	6	278	21	257	23	201	4	195	4
	7	657	122	555	145	887	7	591	11
	8	273	53	279	52	232	7	252	6
	9	250	754	231	815	362	23	271	31
	10	232	599	244	569	250	29	237	31
	11	309	1003	339	914	374	27	305	33
	12	355	543	322	598	491	20	342	28
	13	696	1335	780	1191	1165	15	832	21
	14	161	2706	135	3238	213	51	205	53
	15	294	∞	230	∞	150	∞	145	∞
	16	630	∞	491	∞	1132	∞	635	∞

Tableau C.3 – Valeurs pour une initialisation duale avec $\hat{\pi}_{sr}$

Strat.	Pb.	TL-cpu *MDVSP* stabilisé (s)				Itérations *GENCOL*			
		pb-$\hat{\pi}_{sr}$	rrpt	Dd-$\hat{\pi}_{sr}$	rrdt	pb-$\hat{\pi}_{sr}$	rrpc	dD-$\hat{\pi}_{sr}$	rrdc
Hybride	1	658	1	326	2	832	0	417	1
	2	89	2	89	2	134	1	134	1
	3	195	8	188	8	119	4	117	4
	4	199	4	218	3	168	1	192	1
	5	686	11	669	12	247	5	227	5
	6	562	10	636	9	221	4	213	4
	7	1006	80	755	106	956	7	969	7
	8	243	59	263	55	204	8	205	8
	9	324	581	350	539	475	18	402	21
	10	2950	47	1742	80	981	7	972	7
	11	2002	155	4383	71	1309	8	1979	5
	12	3632	53	2961	65	1401	7	1417	7
	13	696	1335	686	1354	1009	18	561	32
	14	1761	248	1764	247	1011	11	1011	11
	15	3116	∞	3808	∞	6450	∞	4294	∞
	16	854	∞	645	∞	1504	∞	753	∞
d.b.	1	577	1	438	1	1144	0	900	0
	2	34	6	34	6	73	1	73	1
	3	129	12	117	13	146	3	143	3
	4	318	2	328	2	384	1	393	1
	5	364	21	362	21	313	4	309	4
	6	282	21	287	20	223	4	214	4
	7	1550	52	1278	63	1622	4	1341	5
	8	281	51	293	49	275	6	262	6
	9	365	516	412	457	763	11	689	12
	10	225	617	245	567	267	27	243	30
	11	294	1053	292	1061	253	39	240	42
	12	1734	111	1118	172	1116	9	1052	9
	13	1014	916	632	1470	2608	7	795	22
	14	155	2825	173	2520	171	64	180	61
	15	1277	∞	1065	∞	1514	∞	941	∞
	16	751	∞	606	∞	2054	∞	888	∞

Tableau C.4 – Valeurs pour une initialisation duale avec $\hat{\pi}_m$

Strat.	Pb.	TL-cpu *MDVSP* stabilisé (s)				Itérations *GENCOL*			
		pb-$\hat{\pi}_m$	rrpt	Dd-$\hat{\pi}_m$	rrdt	pb-$\hat{\pi}_m$	rrpc	dD-$\hat{\pi}_m$	rrdc
Hybride	1	500	1	177	3	406	1	208	1
	2	82	2	82	2	114	1	114	1
	3	126	12	128	12	112	4	142	3
	4	192	4	198	4	183	1	176	1
	5	244	32	233	33	171	7	169	7
	6	620	9	654	9	235	4	259	3
	7	755	106	538	149	652	10	601	11
	8	856	17	774	19	357	5	347	5
	9	1025	184	1196	158	688	12	651	13
	10	2766	50	1733	80	959	8	948	8
	11	2655	117	2650	117	1968	5	1436	7
	12	3372	57	2840	68	539	18	563	17
	13	214	4340	247	3765	335	53	309	57
	14	1569	278	1571	278	897	12	897	12
	15	390	∞	336	∞	552	∞	405	∞
	16	150	∞	119	∞	289	∞	180	∞
d.b.	1	298	2	263	2	597	0	517	0
	2	28	7	28	7	61	2	61	2
	3	130	12	122	13	165	3	159	3
	4	326	2	385	2	398	1	473	1
	5	282	27	264	29	267	4	245	5
	6	215	27	215	27	183	5	183	5
	7	880	91	965	83	853	8	1083	6
	8	187	77	182	79	158	10	143	11
	9	193	976	210	896	294	28	261	32
	10	160	869	173	804	206	35	235	31
	11	159	1944	155	2002	183	55	147	68
	12	943	204	687	280	563	17	453	21
	13	184	5042	208	4470	310	57	255	70
	14	149	2933	141	3100	219	50	202	54
	15	143	∞	128	∞	117	∞	112	∞
	16	135	∞	107	∞	286	∞	185	∞

Annexe D : Moyennes des critères de performance de la stabilisation par une fonction de pénalité linéaire à quatre morceaux

Tableau D.1 – Valeurs pour une initialisation duale avec $\hat{\pi}_s$

Strat.	Pb.	TL-cpu *MDVSP* stabilisé (s)				Itérations *GENCOL*			
		pb-$\hat{\pi}_s$	rrpt	Dd-$\hat{\pi}_s$	rrdt	pb-$\hat{\pi}_s$	rrpc	dD-$\hat{\pi}_s$	rrdc
Hybride	1	102	5	102	5	152	1	152	1
	2	29	7	31	6	59	2	58	2
	3	111	14	116	13	108	4	124	4
	4	165	4	154	5	179	1	163	2
	5	357	22	362	21	257	4	250	5
	6	1124	5	1820	3	756	1	987	1
	7	528	152	591	136	652	10	615	11
	8	3577	4	3722	4	2354	1	2350	1
	9	7752	24	11277	17	4946	2	5907	1
	10	5288	26	6755	21	5869	1	5055	1
	11	11419	27	16344	19	12030	1	10576	1
	12	7909	24	11626	17	7724	1	7219	1
	13	887	1048	1101	843	1240	14	991	18
	14	13301	33	15383	28	14271	1	14626	1
	15	3211	∞	2195	∞	4841	∞	1910	∞
	16	6602	∞	661	∞	8728	∞	702	∞
d.b.	1	199	3	202	2	395	1	408	1
	2	44	4	42	5	94	1	94	1
	3	184	8	163	9	227	2	207	2
	4	393	2	391	2	490	1	475	1
	5	549	14	542	14	506	2	493	2
	6	290	20	281	21	221	4	219	4
	7	1147	70	881	91	1645	4	1103	6
	8	1228	12	325	44	278	6	291	6
	9	454	415	326	578	578	14	354	23
	10	268	517	299	465	309	23	327	22
	11	357	868	377	822	442	23	368	27
	12	414	465	395	487	556	17	438	22
	13	904	1028	867	1071	1688	11	983	18
	14	252	1731	213	2052	328	33	262	42
	15	392	∞	243	∞	185	∞	158	∞
	16	40653	∞	636	∞	30385	∞	998	∞

Tableau D.2 – Valeurs pour une initialisation duale avec $\hat{\pi}_s$ (suite)

Strat.	Pb.	TL-cpu *MDVSP* stabilisé (s)				Itérations *GENCOL*			
		pb-$\hat{\pi}_s$	rrpt	Dd-$\hat{\pi}_s$	rrdt	pb-$\hat{\pi}_s$	rrpc	dD-$\hat{\pi}_s$	rrdc
d.u.s.	1	272	2	211	2	497	0	386	1
	2	33	6	31	6	64	2	66	2
	3	211	7	198	8	254	2	244	2
	4	336	2	325	2	402	1	382	1
	5	713	11	721	11	615	2	614	2
	6	255	23	240	24	178	5	170	5
	7	523	153	612	131	666	10	671	10
	8	275	52	280	52	227	7	227	7
	9	413	456	360	523	379	22	325	26
	10	248	561	272	510	238	30	261	28
	11	398	779	471	658	401	25	359	28
	12	386	499	384	501	497	20	398	24
	13	687	1352	842	1103	1189	15	883	20
	14	250	1746	217	2012	246	44	219	50
	15	1060	∞	869	∞	821	∞	546	∞
	16	16678	∞	644	∞	18018	∞	996	∞
d.u.f.	1	189	3	173	3	371	1	349	1
	2	35	5	33	6	64	2	66	2
	3	197	8	191	8	247	2	249	2
	4	374	2	358	2	399	1	380	1
	5	663	12	627	12	617	2	564	2
	6	279	21	251	23	183	5	164	5
	7	483	166	520	154	636	10	567	12
	8	281	51	293	49	215	7	224	7
	9	431	438	365	516	404	21	321	26
	10	260	534	281	494	251	29	270	27
	11	397	780	450	688	410	24	360	28
	12	400	481	403	478	485	20	408	24
	13	655	1417	814	1141	1132	16	827	21
	14	249	1753	226	1935	232	47	219	50
	15	950	∞	1064	∞	583	∞	713	∞
	16	15166	∞	643	∞	14638	∞	984	∞

Tableau D.3 – Valeurs pour une initialisation duale avec $\hat{\pi}_{sr}$

Strat.	Pb.	TL-cpu *MDVSP* stabilisé (s)				Itérations *GENCOL*			
		pb-$\hat{\pi}_{sr}$	rrpt	Dd-$\hat{\pi}_{sr}$	rrdt	pb-$\hat{\pi}_{sr}$	rrpc	dD-$\hat{\pi}_{sr}$	rrdc
Hybride	1	551	1	278	2	840	0	412	1
	2	31	6	33	6	60	2	59	2
	3	176	9	181	9	147	3	145	3
	4	180	4	179	4	164	2	160	2
	5	1323	6	1589	5	795	1	778	1
	6	791	7	1027	6	487	2	476	2
	7	2999	27	2644	30	982	7	1205	5
	8	326	44	313	46	322	5	265	6
	9	384	491	443	425	673	12	590	14
	10	6167	23	8838	16	6641	1	6277	1
	11	33929	9	25964	12	31984	0	17696	1
	12	3085	62	2925	66	1592	6	1617	6
	13	722	1286	819	1135	1135	16	763	23
	14	10445	42	10463	42	12600	1	12600	1
	15	8190	∞	4084	∞	14786	∞	4709	∞
	16	11209	∞	760	∞	14403	∞	954	∞
d.b.	1	815	1	484	1	1662	0	998	0
	2	43	4	39	5	89	1	88	1
	3	152	10	135	11	180	3	165	3
	4	369	2	384	2	457	1	471	1
	5	380	20	397	19	327	3	344	3
	6	295	20	318	18	230	4	242	3
	7	3241	25	2709	30	3241	2	2950	2
	8	299	48	321	45	290	6	291	6
	9	504	374	501	376	1195	7	895	9
	10	288	483	292	475	328	22	311	23
	11	330	938	332	933	263	38	294	34
	12	2127	91	1615	119	1211	8	1536	6
	13	1644	565	898	1035	4686	4	1165	15
	14	177	2463	185	2354	234	47	207	53
	15	2408	∞	1359	∞	4583	∞	1329	∞
	16	19399	∞	844	∞	21500	∞	1361	∞

Tableau D.4 – Valeurs pour une initialisation duale avec $\hat{\pi}_{sr}$ (suite)

Strat.	Pb.	TL-cpu *MDVSP* stabilisé (s)				Itérations *GENCOL*			
		pb-$\hat{\pi}_{sr}$	rrpt	Dd-$\hat{\pi}_{sr}$	rrdt	pb-$\hat{\pi}_{sr}$	rrpc	dD-$\hat{\pi}_{sr}$	rrdc
d.u.s.	1	886	1	774	1	1685	0	1470	0
	2	32	6	29	7	60	2	60	2
	3	148	10	127	12	156	3	139	3
	4	297	2	303	2	353	1	358	1
	5	591	13	556	14	481	2	443	3
	6	279	21	300	19	210	4	217	4
	7	4268	19	4202	19	2398	3	2139	3
	8	276	52	294	49	267	6	259	6
	9	354	532	397	475	743	11	595	14
	10	267	521	288	482	262	28	266	27
	11	523	592	490	632	306	33	336	30
	12	1820	106	1753	110	856	11	1608	6
	13	658	1412	721	1288	1254	14	817	22
	14	175	2499	185	2357	191	57	188	58
	15	4943	∞	2610	∞	7184	∞	1724	∞
	16	14060	∞	845	∞	18646	∞	1383	∞
d.u.f.	1	802	1	685	1	1677	0	1454	0
	2	35	6	31	6	62	2	60	2
	3	134	12	127	12	142	3	148	3
	4	332	2	318	2	356	1	340	1
	5	509	15	481	16	435	3	407	3
	6	298	19	314	18	207	4	212	4
	7	4018	20	4050	20	2080	3	2006	3
	8	298	48	311	46	275	6	258	6
	9	355	531	385	490	698	12	548	15
	10	264	525	282	493	234	31	230	32
	11	521	594	459	675	310	32	294	34
	12	1910	101	1845	104	863	11	1683	6
	13	663	1400	708	1313	1289	14	796	22
	14	177	2471	188	2317	199	55	190	57
	15	6410	∞	2565	∞	10459	∞	1781	∞
	16	10376	∞	857	∞	14752	∞	1399	∞

Tableau D.5 – Valeurs pour une initialisation duale avec $\hat{\pi}_m$

Strat.	Pb.	TL-cpu *MDVSP* stabilisé (s)				Itérations *GENCOL*			
		pb-$\hat{\pi}_m$	rrpt	Dd-$\hat{\pi}_m$	rrdt	pb-$\hat{\pi}_m$	rrpc	dD-$\hat{\pi}_m$	rrdc
Hybride	1	318	2	159	3	430	0	195	1
	2	29	7	30	6	57	2	56	2
	3	148	10	173	9	126	4	154	3
	4	161	4	169	4	152	2	152	2
	5	234	33	234	33	174	7	174	7
	6	1016	6	1321	4	653	1	652	1
	7	2355	34	1777	45	601	11	721	9
	8	3956	4	5752	3	3164	1	3005	1
	9	7033	27	10922	17	8747	1	8193	1
	10	3872	36	6472	21	4272	2	4446	2
	11	12887	24	15936	19	13726	1	9962	1
	12	3473	55	3009	64	1811	5	1506	6
	13	317	2934	390	2381	426	42	406	44
	14	12387	35	12254	36	13844	1	13732	1
	15	472	∞	427	∞	742	∞	474	∞
	16	199	∞	170	∞	361	∞	272	∞
d.b.	1	560	1	290	2	1145	0	569	0
	2	38	5	37	5	80	1	82	1
	3	134	11	134	11	173	3	178	3
	4	346	2	373	2	431	1	460	1
	5	313	25	302	26	297	4	289	4
	6	242	24	241	24	206	4	203	4
	7	1999	40	1472	55	2171	3	1558	4
	8	199	72	199	72	165	10	165	10
	9	241	781	232	812	374	22	307	27
	10	203	684	222	626	262	28	248	29
	11	173	1793	185	1677	195	51	180	55
	12	1402	137	790	244	877	11	567	17
	13	287	3234	246	3770	632	28	314	57
	14	176	2487	158	2771	278	39	275	40
	15	184	∞	155	∞	128	∞	133	∞
	16	168	∞	124	∞	376	∞	217	∞

Tableau D.6 – Valeurs pour une initialisation duale avec $\hat{\pi}_m$ (suite)

Strat.	Pb.	TL-cpu *MDVSP* stabilisé (s)				Itérations *GENCOL*			
		pb-$\hat{\pi}_m$	rrpt	Dd-$\hat{\pi}_m$	rrdt	pb-$\hat{\pi}_m$	rrpc	dD-$\hat{\pi}_m$	rrdc
d.u.s.	1	1037	0	425	1	1981	0	780	0
	2	30	6	28	7	59	2	57	2
	3	140	11	136	11	155	3	160	3
	4	30	24	287	3	59	4	341	1
	5	559	14	543	14	495	2	485	2
	6	214	27	212	27	168	5	163	5
	7	2592	31	2543	32	1217	5	1238	5
	8	187	77	196	74	157	10	159	10
	9	247	761	251	751	366	23	312	27
	10	214	649	218	637	168	43	233	31
	11	221	1401	257	1204	218	46	206	48
	12	1455	132	1205	160	480	20	464	21
	13	206	4517	240	3874	350	51	294	60
	14	178	2446	156	2792	276	40	200	55
	15	282	∞	261	∞	238	∞	221	∞
	16	178	∞	118	∞	485	∞	211	∞
d.u.f.	1	906	1	433	1	1930	0	893	0
	2	32	6	29	6	58	2	57	2
	3	119	13	110	14	137	3	136	4
	4	32	22	325	2	58	4	348	1
	5	501	15	443	17	482	2	422	3
	6	235	25	221	26	172	5	159	5
	7	2622	31	2553	31	1355	5	1253	5
	8	199	73	205	70	157	10	157	10
	9	255	738	248	760	370	22	303	27
	10	235	591	224	620	172	42	232	31
	11	225	1376	250	1237	225	44	208	48
	12	1505	128	1252	154	482	20	444	22
	13	204	4554	241	3858	349	51	300	59
	14	181	2416	156	2800	282	39	191	57
	15	255	∞	272	∞	217	∞	243	∞
	16	178	∞	119	∞	401	∞	216	∞

Annexe E : Meilleurs critères de performance de la stabilisation par une fonction linéaire à quatre morceaux

Tableau E.1 – Jeux de paramètres de stabilisation initiaux des meilleurs temps CPU

Op.	Pb.	Stratégie											
		Hybride			d.b.			d.u.s			d.u.f.		
		$\hat{\pi}_m$	$\hat{\pi}_s$	$\hat{\pi}_{sr}$	$\hat{\pi}_m$	$\hat{\pi}_s$	$\hat{\pi}_{sr}$	$\hat{\pi}_m$	$\hat{\pi}_s$	$\hat{\pi}_{sr}$	$\hat{\pi}_m$	$\hat{\pi}_s$	$\hat{\pi}_{sr}$
pb	1	13	12	21	17	12	22	15	17	21	17	18	19
	2	12	18	20	11	12	11	13	22	20	13	21	20
	3	14	16	14	20	19	15	13	22	20	13	21	20
	4	11	11	13	21	13	21	13	22	20	13	21	20
	5	20	14	14	15	18	16	13	22	20	13	21	20
	6	12	11	11	15	19	20	13	22	20	13	21	20
	7	12	21	21	18	15	19	13	22	20	13	21	20
	8	22	15	22	19	11	22	13	22	20	13	21	20
	9	11	13	22	12	16	12	13	22	20	13	21	20
	10	19	11	22	18	20	13	13	22	20	13	21	20
	11	11	12	12	11	11	21	13	22	20	13	21	20
	12	16	15	21	11	19	21	13	22	20	13	21	20
	13	16	15	22	12	14	15	13	22	20	13	21	20
	14	12	22	21	19	15	22	13	22	20	13	21	20
	15	14	22	18	12	17	21	13	22	20	13	21	20
	16	14	21	17	21	22	17	13	22	20	13	21	20
dD	1	19	11	13	18	22	18	11	18	22	20	14	23
	2	18	16	19	19	11	20	15	23	22	15	23	18
	3	11	13	14	12	19	15	13	22	20	13	21	20
	4	11	14	12	17	12	20	13	22	20	13	21	20
	5	13	11	12	15	15	20	13	22	20	13	21	20
	6	11	20	11	16	16	11	13	22	20	13	21	20
	7	13	18	12	16	11	13	13	22	20	13	21	20
	8	14	19	11	16	15	11	13	22	20	13	21	20
	9	14	20	17	17	18	11	13	22	20	13	21	20
	10	19	11	17	18	20	22	13	22	20	13	21	20
	11	12	11	14	17	14	14	13	22	20	13	21	20
	12	11	11	17	18	22	21	13	22	20	13	21	20
	13	17	20	16	12	13	15	13	22	20	13	21	20
	14	12	22	21	15	17	14	13	22	20	13	21	20
	15	21	15	17	11	14	15	13	22	20	13	21	20
	16	16	11	11	14	11	15	13	22	20	13	21	20

Tableau E.2 – Valeurs pour une initialisation duale avec $\hat{\pi}_s$

Strat.	Pb.	TL-cpu *MDVSP* stabilisé (s)				Itérations *GENCOL*			
		pb-$\hat{\pi}_s$	rrpt	Dd-$\hat{\pi}_s$	rrdt	pb-$\hat{\pi}_s$	rrpc	dD-$\hat{\pi}_s$	rrdc
Hybride	1	89	6	95	5	132	2	140	2
	2	28	7	28	7	54	2	54	2
	3	106	15	92	17	102	5	95	5
	4	150	5	148	5	157	2	157	2
	5	319	24	338	23	216	5	234	5
	6	1007	6	1671	3	663	1	894	1
	7	483	166	545	147	607	11	521	13
	8	2876	5	292	50	2001	1	2166	1
	9	5940	32	9285	20	2965	3	3419	2
	10	3602	39	6100	23	4118	2	4588	2
	11	10219	30	14404	22	10852	1	9010	1
	12	7124	27	10351	19	6956	1	6806	1
	13	728	1275	981	947	940	19	872	20
	14	7851	56	10935	40	7283	2	6969	2
	15	2261	∞	1939	∞	3169	∞	1604	∞
	16	1074	∞	523	∞	1920	∞	520	∞
d.b.	1	165	3	168	3	325	1	336	1
	2	34	6	34	6	75	1	75	1
	3	144	11	130	12	167	3	151	3
	4	333	2	317	2	401	1	387	1
	5	486	16	516	15	442	3	470	2
	6	256	23	225	26	180	5	157	5
	7	607	132	560	143	797	8	591	11
	8	279	52	270	54	225	7	224	7
	9	379	498	298	633	367	23	306	27
	10	248	561	261	532	254	29	256	28
	11	321	966	296	1046	371	27	274	36
	12	335	575	347	555	416	23	376	26
	13	651	1427	767	1211	1135	16	832	21
	14	222	1964	200	2181	253	43	203	54
	15	308	∞	225	∞	149	∞	147	∞
	16	1236	∞	522	∞	2700	∞	762	∞

Tableau E.3 – Valeurs pour une initialisation duale avec $\hat{\pi}_s$ (suite)

Strat.	Pb.	TL-cpu *MDVSP* stabilisé (s)				Itérations *GENCOL*			
		pb-$\hat{\pi}_s$	rrpt	Dd-$\hat{\pi}_s$	rrdt	pb-$\hat{\pi}_s$	rrpc	dD-$\hat{\pi}_s$	rrdc
d.u.s.	1	123	4	107	5	208	1	178	1
	2	29	7	26	7	53	2	54	2
	3	119	13	118	13	132	4	135	4
	4	338	2	278	3	353	1	321	1
	5	545	14	615	13	454	2	512	2
	6	231	25	226	26	147	6	154	5
	7	445	180	522	154	538	12	521	13
	8	252	57	250	58	201	8	194	8
	9	326	578	318	593	314	27	277	30
	10	234	594	243	572	217	33	211	34
	11	366	846	389	797	365	27	314	32
	12	333	579	312	617	396	25	303	32
	13	592	1568	760	1222	960	18	754	24
	14	229	1906	206	2117	210	52	196	56
	15	459	∞	440	∞	327	∞	325	∞
	16	647	∞	527	∞	1756	∞	820	∞
d.u.f.	1	92	5	112	5	160	1	213	1
	2	30	6	29	7	54	2	54	2
	3	132	12	121	13	152	3	148	3
	4	338	2	325	2	353	1	322	1
	5	533	15	470	16	477	2	403	3
	6	246	24	231	25	158	5	144	6
	7	422	190	477	168	515	13	486	14
	8	256	56	263	55	191	8	194	8
	9	353	535	319	590	322	26	266	31
	10	238	583	248	561	217	33	218	33
	11	341	908	347	894	332	30	267	37
	12	341	565	323	596	396	25	303	32
	13	547	1699	738	1259	931	19	767	23
	14	229	1904	207	2113	210	52	192	57
	15	395	∞	455	∞	299	∞	336	∞
	16	1217	∞	527	∞	2678	∞	820	∞

Tableau E.4 – Valeurs pour une initialisation duale avec $\hat{\pi}_{sr}$

Strat.	Pb.	TL-cpu *MDVSP* stabilisé (s)				Itérations *GENCOL*			
		pb-$\hat{\pi}_{sr}$	rrpt	Dd-$\hat{\pi}_{sr}$	rrdt	pb-$\hat{\pi}_{sr}$	rrpc	dD-$\hat{\pi}_{sr}$	rrdc
Hybride	1	513	1	269	2	825	0	405	1
	2	31	6	31	6	54	2	54	2
	3	118	13	110	14	116	4	108	4
	4	171	4	168	4	153	2	147	2
	5	1153	7	1285	6	719	2	665	2
	6	608	10	857	7	383	2	410	2
	7	2739	29	2263	35	950	7	903	7
	8	251	57	284	51	220	7	229	7
	9	342	550	413	456	590	14	510	16
	10	5384	26	8238	17	5753	1	5705	1
	11	15516	20	19845	16	15084	1	12887	1
	12	2097	92	2484	78	1262	8	1430	7
	13	616	1508	741	1253	852	21	655	27
	14	8812	50	8853	49	10829	1	10829	1
	15	3459	∞	3582	∞	6561	∞	4268	∞
	16	1007	∞	686	∞	2227	∞	807	∞
d.b.	1	594	1	444	1	1179	0	920	0
	2	32	6	32	6	67	2	67	2
	3	124	12	116	13	140	3	134	4
	4	329	2	319	2	400	1	384	1
	5	305	25	356	22	251	5	304	4
	6	276	21	288	20	205	4	212	4
	7	2086	38	2244	36	1837	4	2310	3
	8	271	53	298	49	264	6	261	6
	9	380	496	419	450	831	10	688	12
	10	245	568	254	548	238	30	246	29
	11	300	1033	305	1014	199	50	12887	173
	12	1966	98	1356	142	1144	8	1062	9
	13	1079	861	719	1292	2689	7	908	20
	14	167	2607	168	2592	176	62	187	58
	15	1491	∞	1122	∞	2040	∞	936	∞
	16	1080	∞	767	∞	2884	∞	1215	∞

Tableau E.5 – Valeurs pour une initialisation duale avec $\hat{\pi}_{sr}$ (suite)

Strat.	Pb.	TL-cpu *MDVSP* stabilisé (s)				Itérations *GENCOL*			
		pb-$\hat{\pi}_{sr}$	rrpt	Dd-$\hat{\pi}_{sr}$	rrdt	pb-$\hat{\pi}_{sr}$	rrpc	dD-$\hat{\pi}_{sr}$	rrdc
d.u.s.	1	628	1	661	1	1159	0	1249	0
	2	29	7	27	7	51	2	55	2
	3	117	13	116	13	117	4	125	4
	4	274	3	261	3	323	1	303	1
	5	518	15	469	16	413	3	357	3
	6	264	22	274	21	189	4	198	4
	7	3432	23	3445	23	1348	5	1369	5
	8	256	56	262	55	237	7	221	7
	9	315	599	369	510	610	14	530	16
	10	244	571	248	560	209	35	205	35
	11	378	820	386	802	276	36	239	42
	12	1702	113	1504	128	798	12	1180	8
	13	541	1716	628	1479	939	19	730	24
	14	160	2723	171	2554	160	68	175	62
	15	2173	∞	2012	∞	1321	∞	1234	∞
	16	1121	∞	784	∞	3312	∞	1270	∞
d.u.f.	1	582	1	575	1	1188	0	1225	0
	2	31	6	29	7	51	2	56	2
	3	121	13	110	14	125	4	119	4
	4	291	2	283	3	305	1	297	1
	5	423	18	371	21	330	3	292	4
	6	267	22	292	20	176	5	190	4
	7	3406	24	3432	23	1531	4	1201	5
	8	272	53	281	51	240	7	233	7
	9	310	608	353	535	570	15	488	17
	10	247	562	262	530	199	36	201	36
	11	353	878	386	803	249	40	237	42
	12	1838	105	1540	125	804	12	1065	9
	13	560	1659	659	1409	917	19	710	25
	14	164	2670	175	2497	166	66	164	67
	15	2249	∞	1900	∞	1177	∞	1147	∞
	16	1130	∞	767	∞	2952	∞	1232	∞

Tableau E.6 – Valeurs pour une initialisation duale avec $\hat{\pi}_m$

Strat.	Pb.	TL-cpu *MDVSP* stabilisé (s)				Itérations *GENCOL*			
		pb-$\hat{\pi}_m$	rrpt	Dd-$\hat{\pi}_m$	rrdt	pb-$\hat{\pi}_m$	rrpc	dD-$\hat{\pi}_m$	rrdc
Hybride	1	286	2	147	3	384	1	182	1
	2	26	7	26	7	49	2	49	2
	3	94	16	90	17	90	5	97	5
	4	145	5	156	5	139	2	141	2
	5	219	35	217	36	156	7	158	7
	6	894	6	1236	5	574	1	584	1
	7	2157	37	1533	52	552	12	598	11
	8	3600	4	4792	3	2782	1	2622	1
	9	5847	32	9840	19	7839	1	7343	1
	10	3673	38	5276	26	3994	2	3658	2
	11	11369	27	14652	21	11867	1	8902	1
	12	3100	62	2654	73	1494	7	1420	7
	13	303	3061	360	2578	364	49	346	51
	14	9550	46	9471	46	11642	1	11535	1
	15	371	∞	345	∞	460	∞	354	∞
	16	173	∞	139	∞	289	∞	186	∞
d.b.	1	473	1	270	2	956	0	528	0
	2	29	7	29	7	62	2	62	2
	3	129	12	122	13	162	3	159	3
	4	275	3	299	2	336	1	360	1
	5	269	29	283	27	254	4	265	4
	6	209	28	207	28	163	5	163	5
	7	1240	65	1032	78	1138	6	1011	7
	8	186	78	176	82	153	11	141	11
	9	218	863	206	917	289	29	247	34
	10	182	765	208	667	226	32	227	32
	11	160	1931	169	1833	169	59	160	62
	12	1257	153	208	925	792	12	227	43
	13	185	5010	213	4367	310	57	259	69
	14	152	2872	140	3109	205	53	202	54
	15	151	∞	137	∞	126	∞	113	∞
	16	136	∞	113	∞	273	∞	185	∞

Tableau E.7 – Valeurs pour une initialisation duale avec $\hat{\pi}_m$ (suite)

Strat.	Pb.	TL-cpu *MDVSP* stabilisé (s)				Itérations *GENCOL*			
		pb-$\hat{\pi}_m$	rrpt	Dd-$\hat{\pi}_m$	rrdt	pb-$\hat{\pi}_m$	rrpc	dD-$\hat{\pi}_m$	rrdc
d.u.s.	1	835	1	330	2	1598	0	594	0
	2	27	7	24	8	49	2	49	2
	3	102	15	99	15	102	5	108	4
	4	27	27	247	3	49	5	288	1
	5	440	18	412	19	379	3	359	3
	6	199	29	189	31	151	6	137	6
	7	2268	35	2259	36	899	7	970	7
	8	168	86	178	81	139	12	140	11
	9	206	916	216	872	319	26	266	31
	10	199	698	201	690	151	48	195	37
	11	191	1626	230	1348	189	53	186	54
	12	1377	140	1142	169	452	22	428	23
	13	189	4914	219	4249	320	55	268	66
	14	147	2963	145	3010	191	57	177	62
	15	222	∞	220	∞	180	∞	187	∞
	16	154	∞	101	∞	404	∞	172	∞
d.u.f.	1	726	1	268	2	1522	0	535	0
	2	29	7	26	7	53	2	49	2
	3	96	16	90	17	102	5	104	5
	4	29	25	299	2	53	5	316	1
	5	387	20	277	28	362	3	246	5
	6	216	27	200	29	151	6	137	6
	7	2321	35	2211	36	936	7	894	7
	8	185	78	191	75	148	11	141	11
	9	215	877	224	841	323	26	253	33
	10	216	644	208	668	151	48	184	39
	11	187	1654	216	1431	176	57	174	57
	12	1383	139	1190	162	458	21	427	23
	13	191	4870	220	4216	302	59	258	69
	14	155	2809	147	2971	191	57	167	65
	15	190	∞	203	∞	160	∞	178	∞
	16	131	∞	102	∞	308	∞	175	∞